Сергей Маклаков
Денис Матвеев

Анализ данных Генератор отчетов Crystal Reports

Санкт-Петербург

«БХВ-Петербург»

2003

УДК 681.3.06
ББК 32.973.26-018,2
Н63

Маклаков С. В., Матвеев Д. В.

Н63 Анализ данных. Генератор отчетов Crystal Reports. — СПб.: БХВ-Петербург, 2003. — 496 с.: ил.

ISBN 5-94157-230-1

Книга представляет собой практическое руководство по созданию аналитических отчетов с помощью генератора отчетов Crystal Reports версии 9. Дан обзор инструментальной среды и принципы выборки, сортировки и группирования данных. Объясняется использование формул, диаграмм и географических карт, матричных отчетов и OLAP-источников данных. Для системных администраторов рассматриваются различные методы распространения отчетов — от экспорта в распространенные форматы до распространения с помощью Crystal Reports Enterprise. Для профессиональных разработчиков информационных систем рассматривается использование Crystal Report Print Engine API и встраиваемых компонентов Crystal Reports (Delphi и Visual Basic), их применение для интеграции в информационные системы, отчетов, созданных в различных средах разработки. В приложениях приведен перечень и подробное описание функций Crystal Report Print Engine API (CRPE), а также описание параметров функций Crystal Report Print Engine API.

Для широкого круга пользователей, интересующихся вопросами анализа данных и подготовки отчетов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Анатолий Адаменко*
Зав. редакцией	*Григорий Добин*
Редактор	*Дмитрий Фунт*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 23.07.03.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 40.
Тираж 2500 экз. Заказ № 1021.
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-230-1

Содержание

Предисловие

Одной из главных задач корпоративных информационных систем, далее — ИС, является оперативное представление информации, необходимой для принятия решений. Вместе с тем структура современных СУБД, на которых основаны ИС, ориентирована в первую очередь на компактное и непротиворечивое хранение информации, а не на оптимизацию выборки данных. Вследствие этого информация, хранящаяся в корпоративных информационных системах, не всегда используется эффективно. Основной проблемой становится не хранение информации, а представление ее конечному пользователю в виде отчета. Традиционно ИС масштаба предприятия имеют ограниченное число встроенных отчетов, прямое назначение которых — представление отчетной информации вышестоящим организациям или государственным службам, а не информационная поддержка руководящего звена. Специфика аналитических отчетов, предназначенных для облегчения процесса принятия решений, состоит в их изменчивости, поскольку в реальной жизни требования бизнеса меняются чуть ли не каждый день. Заказы на разработку таких отчетов поступают разработчикам отдела автоматизации предприятия, если таковой имеется, либо разработчикам сторонних фирм, причем потребности в аналитических отчетах растут по мере их создания в геометрической прогрессии. Рано или поздно разработчик понимает, что решить проблему средствами среды разработки ИС крайне затруднительно. Для этого гораздо лучше подходит специализированный генератор отчетов, подобный Crystal Reports.

Существуют различные подходы к внедрению Crystal Reports как инструмента создания отчета в компании. Наиболее очевидный и, на первый взгляд, простой подход — обучение служащих компании самостоятельному использованию Crystal Reports для создания отчетов. К сожалению, такой путь тяжело реализовать на практике. Дело не в сложности освоения Crystal Reports — навыкам работы в инструментальной среде Crystal Reports можно быстро научить даже слабо подготовленного пользователя. Проблема заключается в источнике данных, с которыми будет работать Crystal Reports. Пользователь в этом случае обращается непосредственно к базе данных и должен понимать, что такое структура данных, и разбираться в соответствии

названий таблиц и полей бизнес-логике. Подавляющее большинство ИС для промышленности и бизнеса имеют в качестве источников данных СУБД западных фирм, для которых локализация далеко не первоочередная задача, поэтому русификация технологической внутренней информации ИС — дополнительная "головная боль" отечественных разработчиков. Следовательно, пользователь Crystal Reports может увидеть только англоязычные "технические" наименования объектов баз данных, что затрудняет работу. Поэтому чаще более эффективным оказывается другой подход — иметь на предприятии специалиста, профессионально занимающегося генерацией отчетов. Такой специалист не обязательно должен быть программистом, но ему необходимо знать предметную область, структуру данных, особенно с точки зрения ее соответствия предметной области, и развитые возможности генератора отчетов. Crystal Reports имеет мощный инструментарий, предназначенный для таких специалистов, позволяющий, например, выполнять сложную статистическую обработку или объединять данные из разнородных источников, например, DBF-файлов и реляционных СУБД.

Третий подход — встраивание отчетов Crystal Reports в клиентские части ИС, созданные при помощи других средств разработки, таких как VB, Power Builder, Delphi, либо публикация отчетов на тонком клиенте с использованием технологии Intranet. В результате пользователь может вызывать и просматривать отчеты в привычной для себя среде. При этом задача создания отчета не снимается, облегчается лишь проблема распространения отчета, поэтому такой подход может служить дополнением, а не альтернативой предыдущим.

Crystal Reports версии 9 предлагается фирмой-производителем Crystal Decisions в четырех вариантах поставки: Standard, Professional, Developer и Advanced. Каждый из вариантов Crystal Reports предназначен для определенного круга пользователей. Наиболее дешевый вариант — Standard представляет собой эффективный инструмент создания отчетов для начинающих, поскольку средства мастера отчетов позволяют работать с Crystal Reports пользователям, не имеющим навыков программирования, и для профессионалов, которым адресован мощный язык программирования. Вариант Standard содержит средства доступа к "настольным" базам данных, но имеет ограниченный набор драйверов доступа к корпоративным СУБД.

Вариант Professional по сравнению с Standard дополнен средствами доступа к данным практически из любого источника, включая XML, OLAP, реляционные базы данных и т. д. Кроме того, вариант Professional включает хранилище для повторно используемых компонентов и расширенный набор средств для просмотра отчетов, в том числе WEB-интерфейс.

В вариант Developer дополнительно входят средства, позволяющие интегрировать отчеты в приложения, созданные в распространенных средах разработки. В состав Developer входит API для создания, просмотра и изменения

отчетов, а также набор инструментальных средств (Java, .NET и COM) для просмотра отчетов.

Advanced представляет собой наиболее дорогой и полный вариант поставки Crystal Reports. По сравнению с вариантом Developer в него включены наборы инструментальных средств (Java, .NET и COM) для создания и изменения отчетов, инструменты доступа к определяемым пользователем источникам данных — .NET, JavaBeans и COM, а также отсутствуют лицензионные ограничения на развертывание отчетов.

В этой книге будут показаны основные возможности среды Crystal Reports релиз 9 применительно к наиболее полному варианту поставки Advanced и практические приемы создания отчетов.

Книга состоит из девяти глав и двух приложений.

Первая глава посвящена обзору инструментальной среды Crystal Reports.

Во *второй главе* рассматриваются приемы создания стандартных отчетов, как с помощью мастера отчетов, так и с использованием инструментальных средств.

В *третьей главе* излагаются принципы создания более сложных отчетов, включающих выборку, сортировку и сложную группировку данных.

Четвертая глава посвящена включению в отчет диаграмм и географических карт.

В *пятой главе* рассматривается создание и использование формул Crystal Reports.

В *шестой главе* описываются приемы сложного форматирования объектов отчетов, в том числе с использованием формул Crystal Reports, и специальные отчеты, как-то: матричные отчеты, отчеты, содержащие подотчеты и отчеты на основе OLAP-источников данных.

Седьмая глава посвящена обзору способов связи Crystal Reports с источниками данных. Особое внимание уделяется работе с SQL-серверами. Дается краткий обзор языка SQL и особенностей его применения в среде Crystal Reports.

В *восьмой главе* рассматриваются различные методы распространения отчетов Crystal Reports — от экспорта в распространенные форматы до распространения с помощью специализированного продукта — Crystal Reports Enterprise.

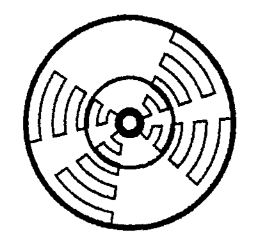
Девятая глава представляет особый интерес для профессиональных разработчиков информационных систем. В ней рассматривается использование Crystal Report Print Engine API (Application Program Interface, интерфейс прикладных программ) и встраиваемых компонентов Crystal Reports (Delphi

и Visual Basic), их использование для интеграции отчетов в информационные системы, созданных в различных средах разработки.

Приложение 1 представляет собой перечень и подробное описание функций Crystal Report Print Engine API (CRPE).

В *приложении 2* приводится описание переменных, используемых при работе с функциями Crystal Report Print Engine API и имеющих тип *запись* или *структура*.

В некоторых главах читателю предлагается сделать упражнения. Во всех примерах и упражнениях, если это не оговорено особо, в качестве источника данных используется файл xtreme.mdb, который после инсталляции Crystal Reports можно найти в каталоге Program Files/Crystal Decisions/Crystal Reports 9/Samples/En/Databases.



Глава 1

Инструментальная среда Crystal Reports 9

1.1. Панели инструментов Crystal Reports 9

Crystal Reports 9 имеет мощную и интуитивно понятную среду разработки отчетов. На рис. 1.1 показано главное окно Crystal Reports 9.

Рис. 1.1. Главное окно Crystal Reports

Системное меню **Control Menu Box** находится в левом верхнем углу, кнопки **Maximize** и **Minimize** — в правом верхнем. В верхней части окна, сразу под

заголовком, находится главное меню Crystal Reports 9 (**Menu bar**). В нижней части окна находится строка состояния **Status Bar**.

В средней части главного окна находится окно просмотра и редактирования отчета с двумя вкладками — **Design** и **Preview**. Режим разработки отчета **Design** предназначен для создания и редактирования отчета, режим просмотра **Preview** — для его просмотра. Crystal Reports 9 позволяет создавать новые объекты и изменять их расположение, размеры и другие свойства как в режиме **Design**, так и в режиме **Preview**. Подробнее о создании и изменении отчетов в режимах **Design** и **Preview** будет рассказано в последующих главах.

Наиболее часто встречающиеся действия вынесены в панели инструментов, которые размещены в верхней части окна. Всего Crystal Reports 9 имеет четыре панели инструментов: главную **Standard**, форматирования **Formatting**, панель инструментов для вставки объектов в отчет **Insert Tools** и панель инструментов для настройки отчетов **Expert Tools**.

Функциональность главной панели инструментов доступна также из основного меню Crystal Reports 9 (табл. 1.1).

Таблица 1.1. *Элементы управления главной панели инструментов Crystal Reports*

Элемент управления	Описание	Соответствующая команда меню
	Создать новый отчет	**File | New...**
	Открыть отчет	**File | Open...**
	Сохранить отчет	**File | Save...**
	Напечатать отчет	**File | Print | Printer...**
	Просмотреть отчет	**File | Print Preview...**
	Экспорт отчета	**File | Export...**
	Обновление отчета	**Report | Refresh Report Data**
	Вырезать (объект)	**Edit | Cut**
	Копировать (объект)	**Edit | Copy**
	Вставить (объект)	**Edit | Paste**

***Таблица 1.1** (окончание)*

Элемент управления	Описание	Соответствующая команда меню
	Отмена действия. Может быть отменено несколько предыдущих действий	**Edit | Undo**
	Повтор действия. Может быть повторено несколько предыдущих действий	**Edit | Redo**
	Включение и выключение окна Group Tree в режиме просмотра отчета	**View | Group Tree**
	Включение и выключение окна Field Explorer	**View | Field Explorer**
	Включение и выключение окна Report Explorer	**View | Report Explorer**
	Включение и выключение окна Repository Explorer	**View | Repository Explorer**
	Поиск в отчете	**Edit | Find...**
100%	Изменение масштаба отчета	**View | Zoom...**
	Вызов контекстной справки	**Help | Context help**

Каждую панель можно показать или скрыть с помощью редактора **Toolbars**, который вызывается командой **View | Toolbars** главного меню (рис. 1.2).

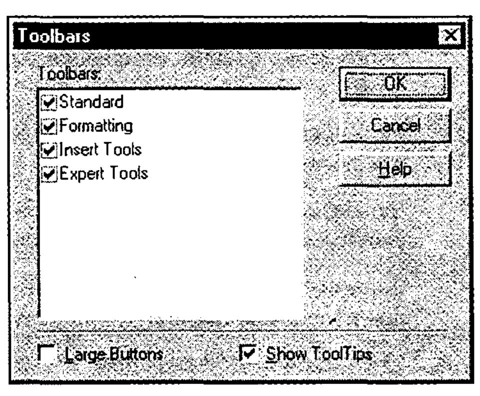


Рис. 1.2. Диалоговое окно **Toolbars**

Любая панель инструментов может быть перемещена в окне программы методом Drag&Drop ("перетащить и отпустить").

Панель форматирования содержит инструменты форматирования объектов отчета и становится доступной только в том случае, когда фокус установлен на соответствующем объекте. Элементы управления панели форматирования приведены в таблице 1.2.

Таблица 1.2. *Элементы управления панели форматирования Crystal Reports 9*

Элемент управления	Описание
Arial	Выбор шрифта для выделенного объекта отчета
10	Выбор размера шрифта для выделенного объекта отчета
	Увеличение и уменьшение шрифта
B I U	Установка полужирного шрифта, курсива и подчеркнутого шрифта
	Форматирование текста объекта по левому краю, центру, правому краю и выравнивание текста соответственно
	Выбор цвета шрифта объекта
	Установка рамок
	Скрытие объекта
	Запрет/разрешение форматирования объекта
	Запрет/разрешение изменения размера и положения объекта
$	Включение/выключение отображения символа валюты для полей типа *Currency*
	Включение/выключение отображения символа разделителя тысяч для полей типа *Currency* и *Number*
%	Включение/выключение отображения символа процента для полей типа *Number*
	Уменьшение и увеличение отображаемого количества знаков после запятой для полей типа *Currency* и *Number*

Для вставки в отчет новых объектов, например текстовых объектов, групп или суммирующих полей служит специальная панель вставки (табл. 1.3).

Таблица 1.3. *Элементы управления панели инструментов для вставки объектов в отчет*

Элемент управления	**Описание**	**Соответствующая команда меню**
	Вставить текстовый объект	**Insert | Text Object**
	Вставить группу	**Insert | Group...**
	Вставить суммирующее поле	**Insert | Summary...**
	Вставить объект Cross-Tab (матричный отчет)	**Insert | Cross-Tab...**
	Вставить объект OLAP Grid (матричный отчет на основе OLAP-источника данных)	**Insert | OLAP Grid...**
	Вставить подотчет	**Insert | Subreport...**
	Нарисовать линию	**Insert | Line**
	Нарисовать прямоугольник	**Insert | Box**
	Вставить изображение	**Insert | Picture...**
	Вставить диаграмму	**Insert | Chart...**
	Вставить географическую карту	**Insert | Map...**

Панель инструментов для настройки отчетов (табл. 1.4) позволяет вызвать диалоговые окна, с помощью которых можно установить сортировку данных отчета, произвести выборку и группировку данных, связать таблицы базы данных и изменить свойства любого объекта отчета. Подробнее работа с такими диалоговыми окнами будет рассмотрена в последующих главах книги.

Таблица 1.4. Элементы управления панели инструментов для анализа

Элемент управления	Описание	Соответствующая команда меню
	Вызов диалогового окна **Database Expert**, с помощью которого можно связать данные из таблиц баз данных	**Database | Database Expert...**
	Вызов диалогового окна **Group Expert** для изменения порядка группировки и свойств групп	**Report | Group Expert...**
	Вызов диалогового окна **Group Sort Expert** для специального порядка группировки, например, группировки Top N	**Report | Group Sort Expert...**
	Вызов диалогового окна **Record Sort Expert** для сортировки строк отчета	**Report | Record Sort Expert...**
	Вызов диалогового окна **Select Expert** для создания и изменения правил выборки данных	**Report | Select Expert...**
	Вызов диалогового окна **Section Expert** для форматирования секций отчета	**Report | Section Expert...**
	Вызов диалогового окна **Formula Workshop** для доступа к хранилищу формул	**Report | Formula Workshop...**
	Вызов диалогового окна **OLAP Report Creation Wizard**	**Report | OLAP Report Setting...**
	Вызов диалогового окна — редактора шаблонов отчетов **Template Expert**	**Report | Template Expert...**
	Вызов диалогового окна — редактора свойств объектов отчета **Format Editor**	**Format | Format Editor...**
	Создание гиперссылки для объекта отчета	**Format | Hyperlink...**
	Вызов диалогового окна **Highlighting Expert** для выделения объекта отчета по условию, например, цветом	**Format | Highlighting Expert...**

1.2. Редактирование объектов отчета

В левой части главного окна располагаются три окна, предназначенные для создания, удаления и изменения объектов отчета — **Report Explorer**, **Field Explorer** и **Repository Explorer**. Следует отметить, что все три окна являются перемещаемыми и их можно расположить в любом месте главного окна Crystal Reports 9.

Окно **Report Explorer** (рис. 1.3) содержит древовидный список объектов отчета. Верхний уровень дерева представляет собой отчет как таковой. На втором уровне показаны секции отчета, на следующем показываются подсекции, если отчет содержит множественные секции. Подробнее работа с секциями, в том числе с множественными секциями будет рассмотрена в *главе 6*. Для каждой секции на нижнем уровне показываются содержащиеся в ней объекты.

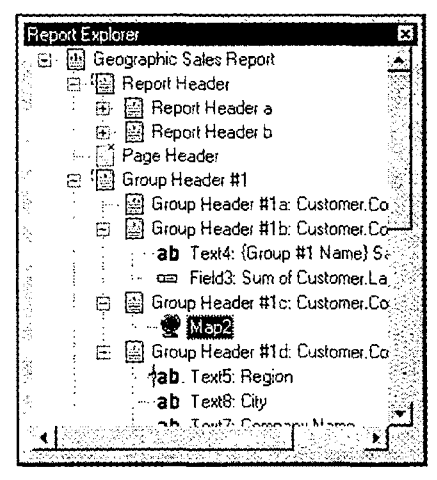


Рис. 1.3. Диалоговое окно **Report Explorer**

Report Explorer позволяет найти в отчете любой объект и изменить его свойства. Если переместить указатель на объект в списке **Report Explorer** и щелкнуть кнопкой мыши, то фокус автоматически установится на этом объекте. Это верно как в режиме **Design**, так и в режиме **Preview**. Для редактирования свойств объекта следует щелкнуть на нем правой кнопкой мыши — при этом появляется контекстное меню. Report Explorer позволяет также редактировать свойства группы объектов. Для выделения группы объектов следует щелкнуть на них кнопкой мыши, одновременно нажимая клавиши <Ctrl> или <Shift>.

Окно **Field Explorer** (рис. 1.4) также служит для редактирования объектов отчета, но в нем объекты сгруппированы не по размещению в секциях, а по

типу. В отличие от **Report Explorer**, в окне **Field Explorer** можно создавать новые объекты.

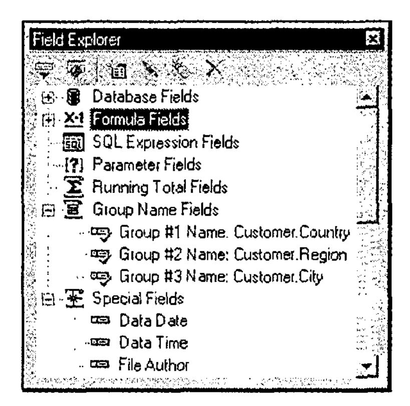


Рис. 1.4. Диалоговое окно **Field Explorer**

С помощью **Field Explorer** можно не только изменять свойства объектов, но и вносить их в любую секцию отчета. Подробнее работа с диалоговым окном **Field Explorer** будет рассмотрена в *главе 2*.

Объекты отчета, например, запросы, функции, определенные пользователем, или текстовые объекты могут иметь сложную структуру, и их создание иногда требует значительных трудозатрат. Очевидно, что такие объекты целесообразно использовать повторно. Crystal Reports 9 имеет специальное средство для повторного использования объектов в нескольких отчетах — хранилище объектов. Окно **Repository Explorer** (рис. 1.5) служит для управления хранилищем объектов — для размещения объектов, их организации в папках хранилища, вставки объектов хранилища в отчет и т. д.

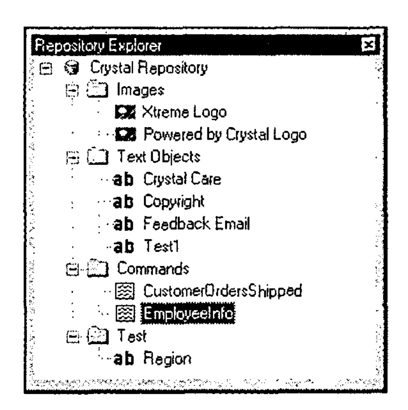


Рис. 1.5. Диалоговое окно **Repository Explorer**

Подробнее работа с хранилищем объектов будет рассмотрена в *разделе 2.10*.

1.3. Использование подсказок

Crystal Reports 9 имеет мощную систему контекстной справки (помощи). Для вызова помощи необходимо выполнить команду главного меню **Help | Crystal Reports Help** или нажать клавишу <F1>. Появляется диалоговое окно, содержащее текст подсказки (рис. 1.6).

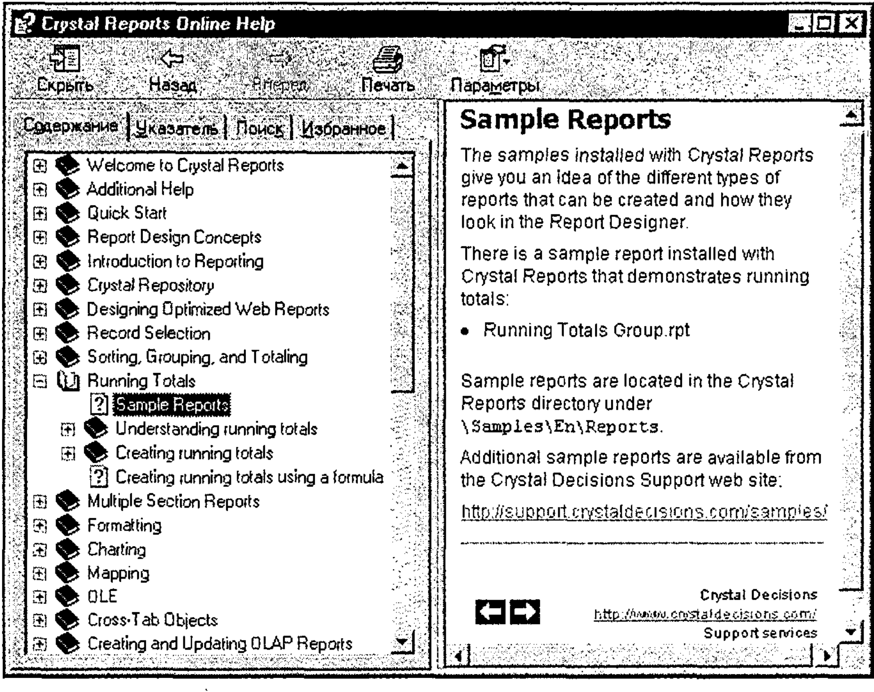
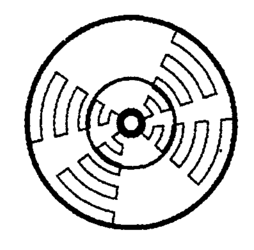


Рис. 1.6. Диалоговое окно **Crystal Reports Online Help**

Для поиска информации можно использовать индексированный список — вкладка **Index** (Указатель), поиск — вкладка **Search** (Поиск) или гиперссылки. Контекстная подсказка вызывается нажатием клавиш <Shift>+<F1>.

По команде главного меню **Hepl | About Crystal Reports** дается подробная информация о версии Crystal Reports.

Глава 2



Создание стандартных отчетов

2.1. Создание отчета с помощью Standart Report Creation Wizard

Crystal Reports 9 содержит средства, позволяющие создавать отчеты пользователям, не являющимся специалистами в области информационных технологий. Наиболее просто создание отчета с помощью мастера отчетов **Report Creation Wizard**. Мастер отчетов содержит несколько окон, в которых пользователю предлагается создать отчет в несколько шагов.

1. Выбрать тип отчета.
2. Выбрать источник данных для отчета.
3. Связать таблицы источника данных.
4. Создать поля отчета.
5. Сгруппировать данные.
6. Создать суммирующие поля для каждой группы.
7. Сортировать данные отчета.
8. Создать диаграммы на основании данных.
9. Установить правила выборки данных.
10. Выбрать шаблон форматирования полей отчета.

Следовательно, мастер отчетов позволяет создавать сложные отчеты, которые объединяют данные из разных таблиц баз данных, фильтруют данные по заданным критериям, группируют и сортируют их. Для создания нового отчета необходимо выполнить команду меню **File | New** или щелкнуть на кнопке создания нового отчета (табл. 1.1). Открывается диалоговое окно **Crystal Reports Gallery** (рис. 2.1).

Первый шаг — выбор типа нового отчета. В диалоговом окне **Crystal Reports Gallery** можно выбрать один из четырех типов: **Standard**, **Cross-Tab**, **Mail Label** или **OLAP**. Далее будет рассмотрено создание стандартного отчета

Standard. Другие типы отчетов — матричный отчет **Cross-Tab**, отчет для печати этикеток **Mail Label** и отчет на основе OLAP-источников данных **OLAP** будут рассмотрены в *главе 6*. После выбора типа создаваемого отчета **Standard** и щелчка на кнопке **OK** в диалоговом окне **Crystal Reports Gallery** открывается мастер **Standard Report Creation Wizard**.

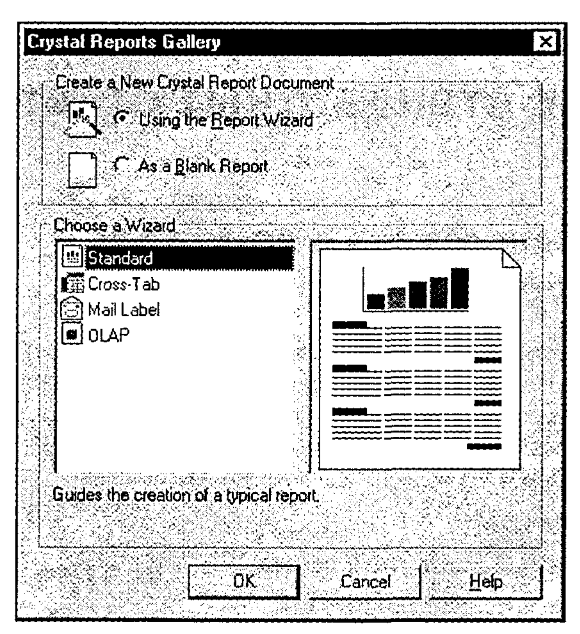


Рис. 2.1. Диалоговое окно **Crystal Reports Gallery**

Standard Report Creation Wizard содержит несколько последовательно открывающихся окон, которые позволяют шаг за шагом создать стандартный отчет. Для перехода к следующему окну в мастере отчетов следует щелкнуть на кнопке **Next**, для возврата к предыдущему — на кнопке **Back**. Если для создания отчета достаточно информации, мастер отчетов делает доступной кнопку **Finish**. Щелчок на кнопке **Finish** закрывает мастер отчетов и открывает новый отчет в режиме просмотра.

Первое окно мастера **Data** (рис. 2.2) содержит два древовидных списка. В левом списке **Available Data Sources** перечислены источники данных, которые могут быть использованы для создания отчета. Доступ к базе данных может быть осуществлен с помощью ODBC или драйверов прямого доступа. Правый список **Selected Tables** содержит выбранные таблицы источников данных.

Рассмотрим создание отчета на основе базы данных, входящей в поставку Crystal Reports 9. Файл базы данных xtreme.mdb после инсталляции Crystal Reports 9 находится в каталоге Program Files\Crystal Decisions\Crystal Reports 9\

Samples\En\Databases. Для выбора xtreme.mdb в качестве источника данных следует переместить указатель на строку **Create New Connection/Database Files/Find Database Files** левого списка и дважды щелкнуть кнопкой мыши. В появившемся диалоговом окне **Open** следует выбрать файл xtreme.mdb и щелкнуть на кнопке **Open**. Затем необходимо переместить указатель на строку с наименованием таблицы и щелкнуть кнопкой мыши на кнопке >. Таблица будет внесена в правый список.

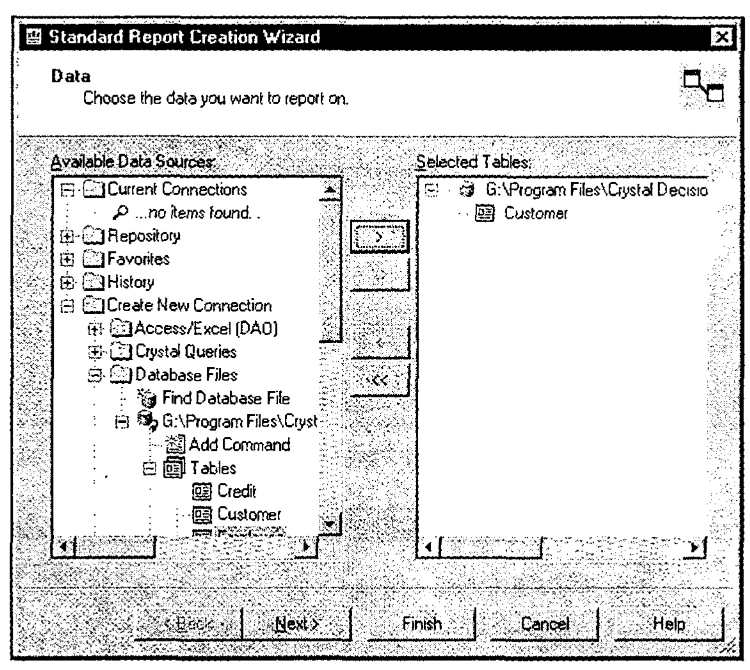


Рис. 2.2. Окно **Data** мастера **Standard Report Creation Wizard**

Щелчок на кнопке **Next** открывает второе окно мастера **Standart Report Creation Wizard (Link)** (рис. 2.3). Если в качестве источника данных было выбрано более одной таблицы, в окне **Link** показываются таблицы отчета в графическом виде — таблица изображается в виде прямоугольника, внутри которого содержится список полей таблицы. Индексированные поля помечены цветными маркерами. По умолчанию таблицы автоматически связываются по полям с одинаковыми названиями. Связи таблиц можно удалять, создавать и менять их свойства. Подробнее связывание таблиц будет рассмотрено в *главе 7*. Заметим, что если в качестве источника данных выбрана "настольная" база данных (например, формата DBF или, как в рассматриваемом случае, база MS Access), то связь между таблицами возможна только по индексированным полям. Для ODBC-источников данных такого ограничения нет.

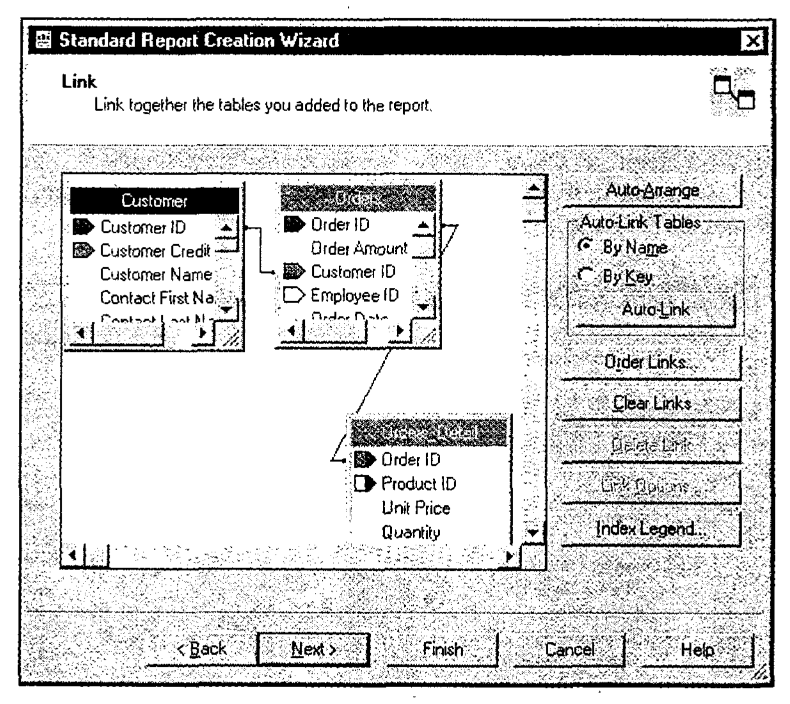


Рис. 2.3. Окно **Link** мастера **Standard Report Creation Wizard**

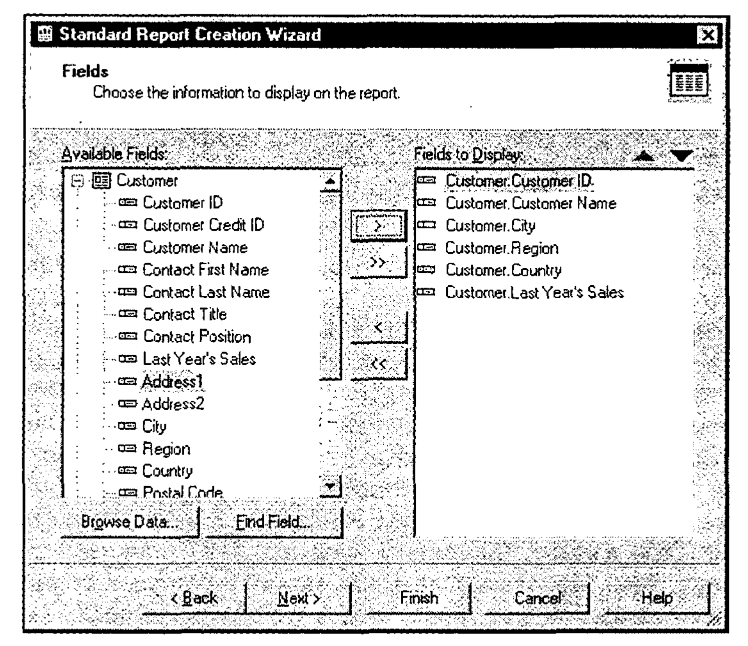


Рис. 2.4. Окно **Fields** мастера **Standard Report Creation Wizard**

Третье окно мастера **Standard Report Creation Wizard** (**Fields**) содержит два древовидных списка (рис. 2.4). Левый список **Available Fields** содержит поля таблиц, на основе которых могут быть созданы объекты отчета — поля (**Fields**) базы данных. Правый список **Fields to Display** содержит поля, которые будут включены в отчет. Наименование поля в правом списке состоит из двух частей, разделенных точкой. Слева от точки показывается наименование таблицы, справа — наименование поля, например, **Customer.Last Year's Sales**. Для включения поля в отчет необходимо переместить указатель на строку с наименованием колонки и щелкнуть на кнопке >.

Окно **Grouping** (рис. 2.5) позволяет сгруппировать данные по какому-либо полю, причем сортировка групп может быть установлена по возрастанию значения поля (числового, строкового или даты), по убыванию или в специальном порядке.

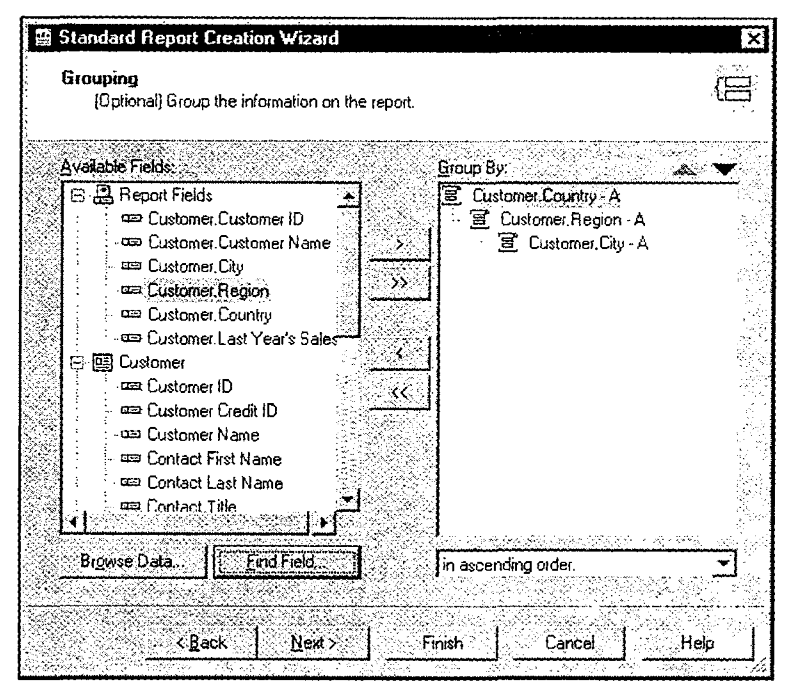


Рис. 2.5. Окно **Grouping** мастера **Standard Report Creation Wizard**

Левый список окна **Grouping** — **Available Fields** содержит поля отчета и поля таблиц, на основе которых могут быть созданы группы отчета. Правый список **Group By** содержит группы отчета. Для создания в отчете новой группы необходимо в левом списке переместить указатель на строку с наименованием поля отчета, по которому будет проведена группировка, и щелкнуть кнопкой >. В рассматриваемом примере (рис. 2.5) данные отчета будут

сгруппированы сначала по странам, внутри каждой страны по регионам и, наконец, внутри каждого региона по городам.

Подробнее группирование данных отчета будет рассмотрено в *главе 3*.

В Crystal Reports 9 можно вычислять суммарные значения числовых полей в каждой группе. Например, если в отчет включено поле **Customer.Last Year's Sales** (продажи клиенту в прошлом году) и в отчете созданы группы по странам, регионам и городам, можно вычислить объемы продаж в каждом городе, регионе и стране. Вычисляться могут не только сумма, но и среднее значение, количество клиентов и другие агрегатные значения. Для вычисления агрегатных значений служит окно **Summaries** (рис. 2.6).

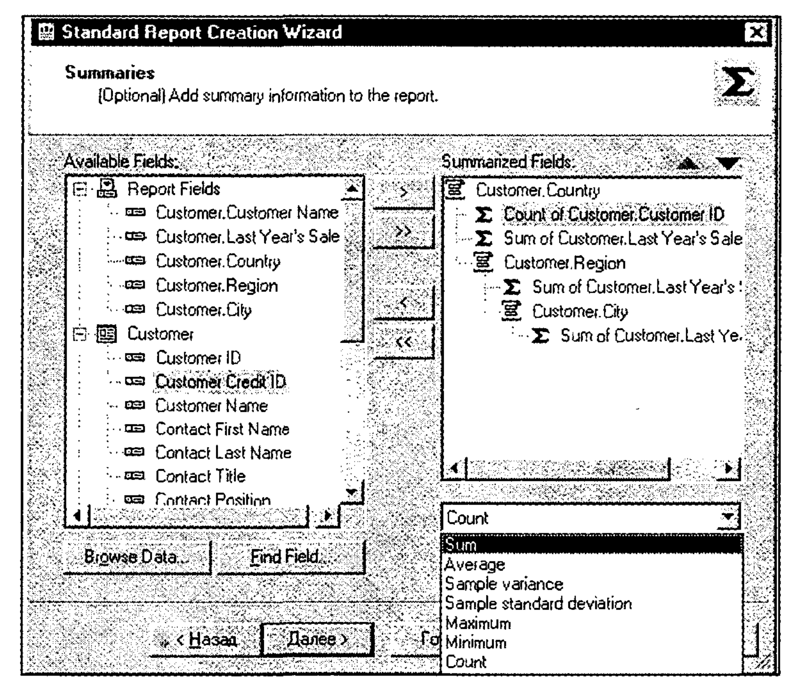


Рис. 2.6. Окно **Summaries** мастера **Standard Report Creation Wizard**

Левый список окна **Summaries** содержит поля отчета и поля таблиц, на основе которых в отчете может быть проведено агрегирование данных. Правый список **Summarized Fields** содержит древовидный список отобранных для вычисления агрегатных значений полей. В правой нижней части окна расположен раскрывающийся список агрегатных функций (суммирование, вычисление среднего, максимального и минимального значения, подсчет количества записей и т. д.). В примере на рис. 2.6 будет вычисляться количество клиентов в каждой стране и сумма продаж в каждой стране, регионе и городе.

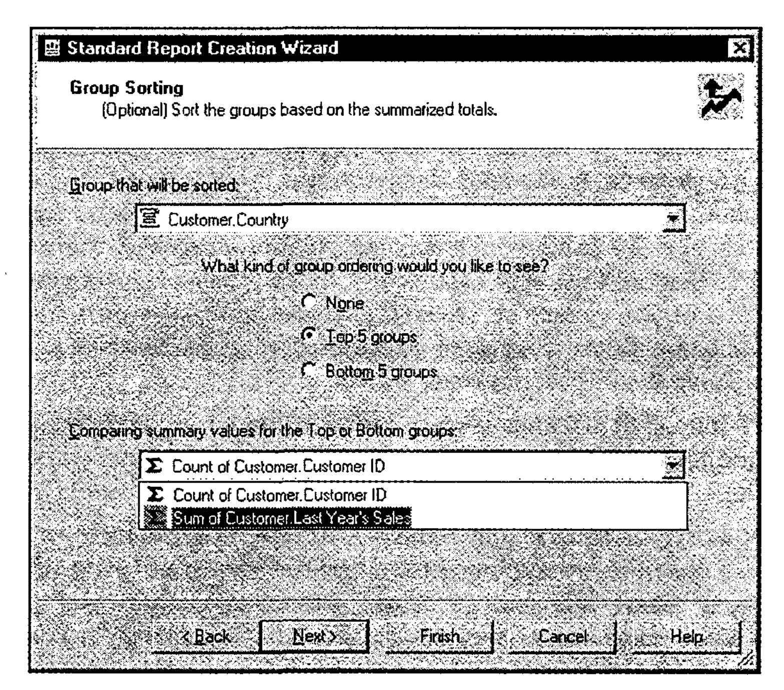


Рис. 2.7. Окно **Group Sorting** мастера **Standard Report Creation Wizard**

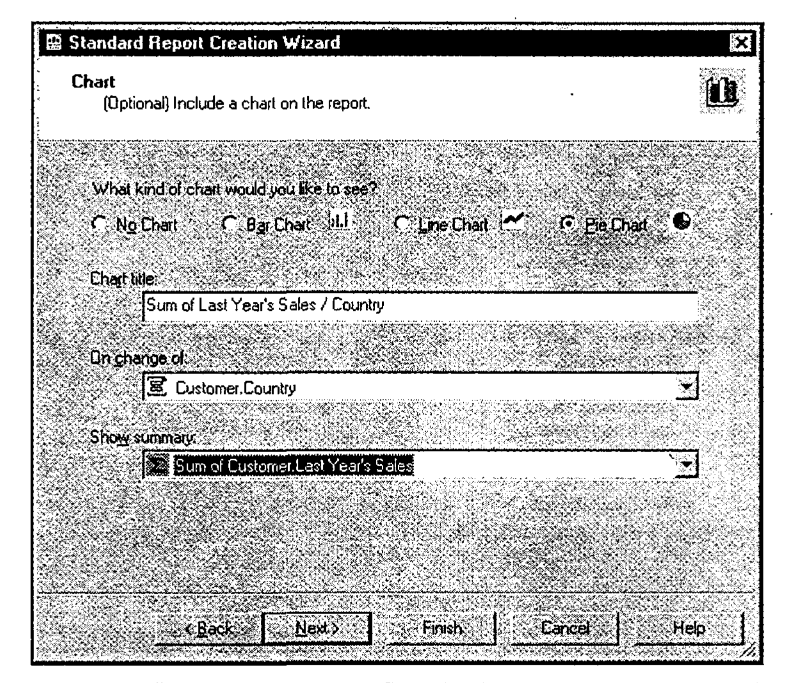


Рис. 2.8. Окно **Chart** мастера **Standard Report Creation Wizard**

Окно **Group Sorting** мастера **Standard Report Creation Wizard** (рис. 2.7) позволяет более эффективно обрабатывать сгруппированные данные. В верхней части окна расположен раскрывающийся список, содержащий группы, уже созданные в отчете с помощью окна **Grouping** (рис. 2.5). Радиокнопки в средней части окна позволяют выбрать тип группировки, а раскрывающийся список в нижней части — выбрать критерий отбора групп. Так, в примере на рис. 2.7 выбор специальной группировки **Top 5 groups** и значения суммы **Sum of Customer.Last Year's Sales** в качестве критерия позволит оставить в отчете только те страны, которые занимают пять первых мест по объему продаж. Более подробно возможности агрегирования и специальной группировки будут рассмотрены в *главе 3*.

Окно **Chart** (рис. 2.8) предназначено для включения в отчет диаграмм трех типов — диаграммы в виде столбцов (**Bar**), линий (**Line**) и секторные диаграммы (**Pie**).

Диаграммы могут быть созданы только в отчете, содержащем группировку данных и вычисление агрегированных значений для этих групп. Окно **Chart** содержит поле **Chart title** для внесения заголовка диаграммы, списки **On change of** и **Show summary**, в которых следует выбрать группу и агрегированные данные для диаграммы.

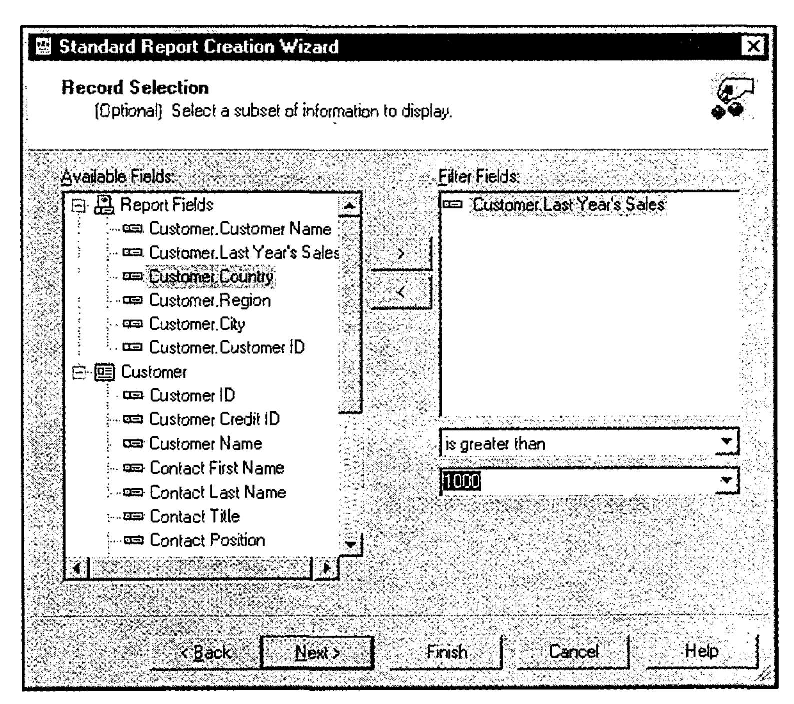


Рис. 2.9. Окно **Record Selection** мастера **Standard Report Creation Wizard**

Окно **Record Selection** (рис. 2.9) служит для создания правил отбора данных в отчет и содержит два списка. Левый древовидный список **Available Fields** содержит все доступные поля таблиц базы данных. Правый список **Filter Fields** содержит поля, на основе которых создаются правила отбора данных в отчет. Для включения поля в правый список из левого необходимо переместить указатель на строку с наименованием колонки и щелкнуть на кнопке >. Например, пусть в отчет следует внести не всех клиентов, а только тех, кто в прошлом году сделал покупок более чем на 1000$. Для этого надо перенести поле **Last Year's Sales** из левого списка в правый и в раскрывающихся списках в правой нижней части окна задать логическое условие — предикат is greater then (больше чем) и числовое значение 1000 (рис. 2.9). Если строка удовлетворяет заданным условиям, она включается в отчет. Предикаты, установленные для различных полей, объединяются логическим "и". Список выбора в правой нижней части окна контекстный, т. е. его содержание — логические условия — зависит от типа выбранного поля. Более сложные условия отбора данных можно установить в диалоговом окне **Select Expert**, которое будет рассмотрено в *главе 3*.

Последнее окно мастера **Standard Report Creation Wizard** — **Template** служит для выбора шаблона форматирования отчета. Список в окне **Template** содержит десять шаблонов. В примере на рис. 2.10 выбран шаблон **Confidential Underlay**.

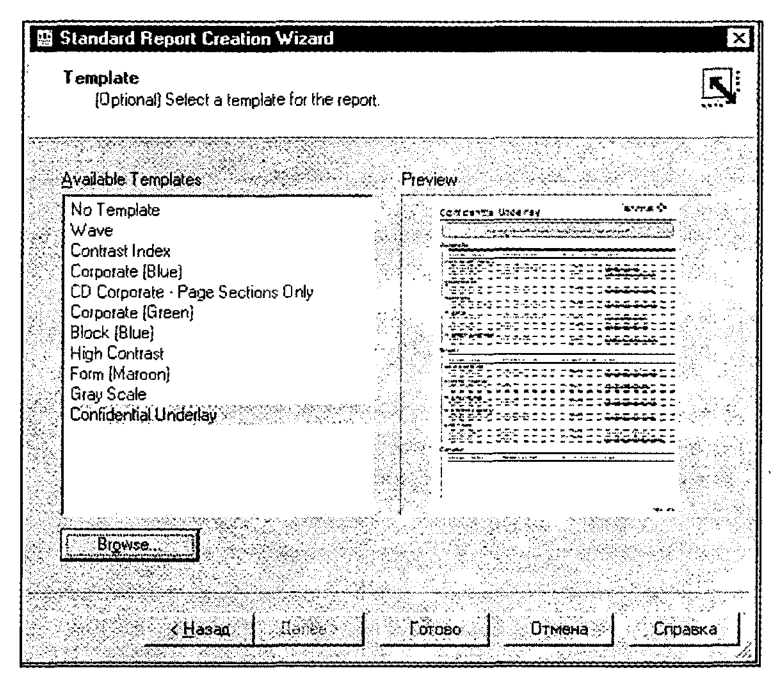


Рис. 2.10. Окно **Template** мастера **Standard Report Creation Wizard**

После щелчка на кнопке **Finish** (Готово) можно просмотреть созданный отчет на вкладке **Preview** главного окна Crystal Reports 9 (рис. 2.11).

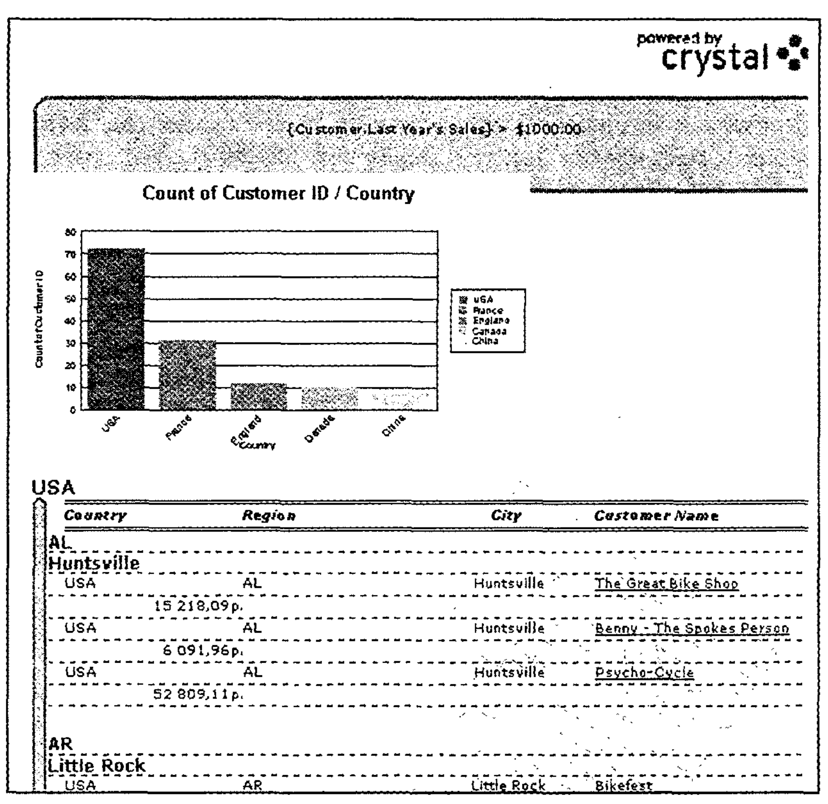


Рис. 2.11. Просмотр на вкладке **Preview** отчета, созданного с помощью мастера **Standard Report Creation Wizard**

Упражнение 2.1

1. Выполните команду меню **File | New**.
2. Выберите в диалоге **Crystal Report Gallery** стандартный отчет **Standard**.
3. В окне **Data** мастера создания отчетов в качестве источника данных выберите **Create New Connection/ Database Files**.
4. В диалоге **Open** (открытие файла) укажите файл Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Databases\xtreme.mdb.
5. После включения базы данных xtreme в список источников выберите в разделе **Tables** таблицу **Customer** и щелкните на кнопке >. Таблица **Customer** включается в правый список **Selected Tables**. Щелкните на кнопке **Next** (Далее).

6. В окне **Fields** включите в отчет поля **Customer Name** и **Last Year's Sales**. Щелкните на кнопке **Next**.
7. В окне **Grouping** включите в отчет группировку по полю **City**.
8. Щелкните несколько раз на кнопке **Next** до тех пор, пока не появится окно **Record Selections**.
9. В окне **Record Selections** задайте условие отбора **Country is equal to USA**.
10. Щелкните на кнопке **Finish**.
11. Сохраните отчет.

2.2. Вставка в отчет и форматирование полей базы данных

Рассмотрим, как выглядит отчет на вкладке **Design**. Большая белая область в середине вкладки разделена на секции горизонтальными линиями (рис. 2.12). При добавлении секции в отчет, например, при группировке данных, Crystal Reports 9 автоматически добавляет линию. Серая область слева содержит дополнительную информацию, помогающую работать с данными и объектами. Горизонтальные линии продолжаются в серую область, определяя секции, и Crystal Reports 9 идентифицирует каждую секцию по аббревиатуре или выбранному имени.

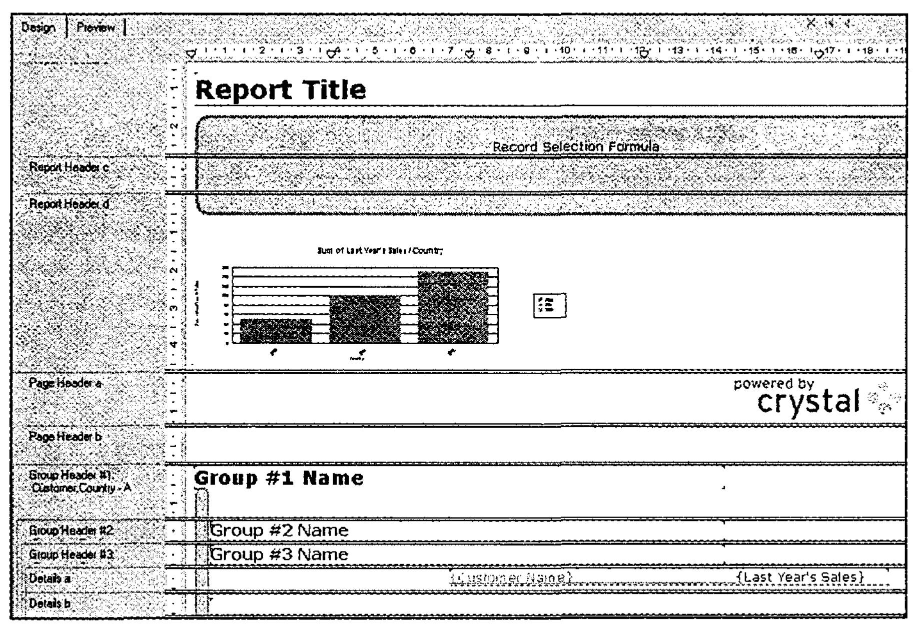


Рис. 2.12. Представление отчета на вкладке **Design**

Секция заголовка отчета Report Header (RH) изображается единожды в самом начале отчета. Секции Page Header (PH) и Page Footer (PF) показываются на каждой странице и обычно используются для заголовков, нумерации страниц и т. д. Секция Detail (D) — это основное содержание отчета. Секция Report Footer (RF) показывается единожды в самом конце отчета.

Для внесения в отчет полей базы данных, специальных полей, формул и параметров служит диалоговое окно **Field Explorer** (рис. 1.4), содержащее панель управления и древовидный список объектов отчета. Элементы управления панели инструментов диалогового окна **Field Explorer** приведены в табл. 2.1.

Таблица 2.1. *Элементы управления панели инструментов диалогового окна* ***Field Explorer***

Элемент управления	**Описание элемента**
	Внесение выбранного объекта в отчет
	Просмотр содержимого поля базы данных (первые 100 неповторяющихся значений)
	Создание объекта
	Редактирование объекта
	Переименование объекта
	Удаление объекта

При внесении в отчет поля базы данных (раздел списка **Database Fields**) доступны только первые две кнопки. Для внесения поля в отчет нужно выбрать поле в списке, щелкнуть на кнопке внесения объекта и затем щелкнуть на свободной части какой-либо секции отчета. Можно также перенести поле из списка в секцию отчета методом Drag&Drop.

Существует три варианта выбора поля для отчета.

- Выбрать индивидуально поле, щелкнув на нем, и поместить его в нужное место отчета.
- Выбрать диапазон полей, щелкнув на первом поле и щелкнув на последнем при нажатой клавише <Shift>.
- Выбрать группу полей, щелкнув на первом поле и щелкнув на каждом дополнительном при нажатой клавише <Ctrl>.

Поля будут размещены в порядке списка, но не в порядке выбора. Размер поля в отчете зависит от размера поля в базе данных: Если поле размещено в секции **Detail**, в секцию **Page Header** одновременно вносится текстовый объект — заголовок поля, который представляет собой название поля в базе данных.

Упражнение 2.2

1. Откройте отчет, созданный в упражнении 2.1.
2. Внесите в секцию отчета **Detail** следующие поля таблицы **Customer**:
   - Contact Last Name
   - Contact First Name
   - Contact Title
3. Сохраните отчет.

Для просмотра полученного отчета на вкладке **Preview** следует щелкнуть на кнопке просмотра отчета (табл. 1.1) на панели инструментов. Отметим, что строка состояния **Status bar** в режиме **Preview** показывает количество выбранных и общее число прочитанных записей.

Crystal Reports 9 позволяет изменить порядок расположения полей отчета. Для этого можно просто перенести поле внутри секции или между секциями методом Drag&Drop. Можно также сразу перенести группу полей. Для этого нужно предварительно выбрать их, щелкнув в каждом, одновременно нажимая клавишу <Shift> или <Ctrl>.

Для более тщательного форматирования полей предназначается специальный инструмент **Guidelines** (опорные линии), к которым можно привязать поля отчета. Для отображения **Guidelines** нужно выполнить команду меню **Files | Options** и на вкладке **Layout** диалогового окна **Options** включить опции **Guidelines**.

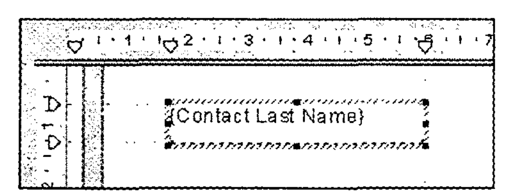


Рис. 2.13. Форматирование полей с помощью опорных линий **Guidelines**

При форматировании поля его левый край подводится к опорной линии (а не наоборот!). То, что поле связано с опорной линией, показывают небольшие красные метки по периметру поля. Если с одной линией связаны несколько полей, то перемещение опорной линии в верхней или левой линейке форматирования (рис. 2.13) приводит к перемешению группы полей

по горизонтали или по вертикали соответственно. Если привязать правый край поля, то с помощью опорной линии можно изменить его размер.

Создать новую опорную линию можно, дважды щелкнув по линейке форматирования. Удалить опорную линию можно, переместив ее (Drag&Drop) из линейки форматирования.

Упражнение 2.3

1. Откройте отчет, созданный в упражнении 2.2.
2. Передвиньте поле и заголовок поля **City** одновременно.
3. Щелкните по полю **City** и выделите его.
4. Нажмите клавишу <Ctrl> и щелкните по заголовку этого поля. Оба поля теперь имеют рамку.
5. Перенесите **City** в левую часть отчета.
6. Сохраните отчет.

Таким же способом можно передвигать любые другие поля. На рис. 2.14 показан отчет, который должен получиться после выполнения упражнения.

21.04.03

City	Customer Name	Last Year's Sales	Contact Title	Contact First	Contact Last
Atlanta					
Atlanta	Tony's Better Bikes	2 735,25р.	Mr.	Anthony	Weatherhead
Atlanta		**2 735,25 р.**			
Austin					
Austin	Rockshocks for Jocks	40 778,52р.	Ms	Heather	Davis
Austin		**40 778,52 р.**			

Рис. 2.14. Представление отчета на вкладке **Preview** после выполнения упражнения 2.3

Если создать отчет и затем сохранить или закрыть его, Crystal Reports 9 по умолчанию вместе с ним сохраняет данные. Если после этого открыть отчет, он будет содержать сохраненные данные. Для принудительного обновления данных следует выполнить команду меню **Report | Refresh Report Data** либо нажать клавишу <F5>.

Для форматирования поля служит диалоговое окно **Format Editor** (рис. 2.15), которое можно вызвать, щелкнув правой кнопкой мыши на поле и выбрав в контекстном меню пункт **Format Field**. Вкладки диалогового окна **Format Editor** позволяют задавать свойства полей безусловно (поля выбора) или по условию (кнопки вызова редактора формул справа от каждого условия).

С помощью редактора формул можно создать формулу, возвращающую логическое значение "истина" или "ложь". При выполнении условия (значение "истина") форматирование будет выполняться, в противном случае — нет. Синтаксис формул будет рассмотрен в *главе 5*. Так, опция **Suppress** позволяет скрыть поле. Например, задав формулу, можно скрывать только поля, содержащие значения меньше 1000. Создание формул будет рассмотрено далее.

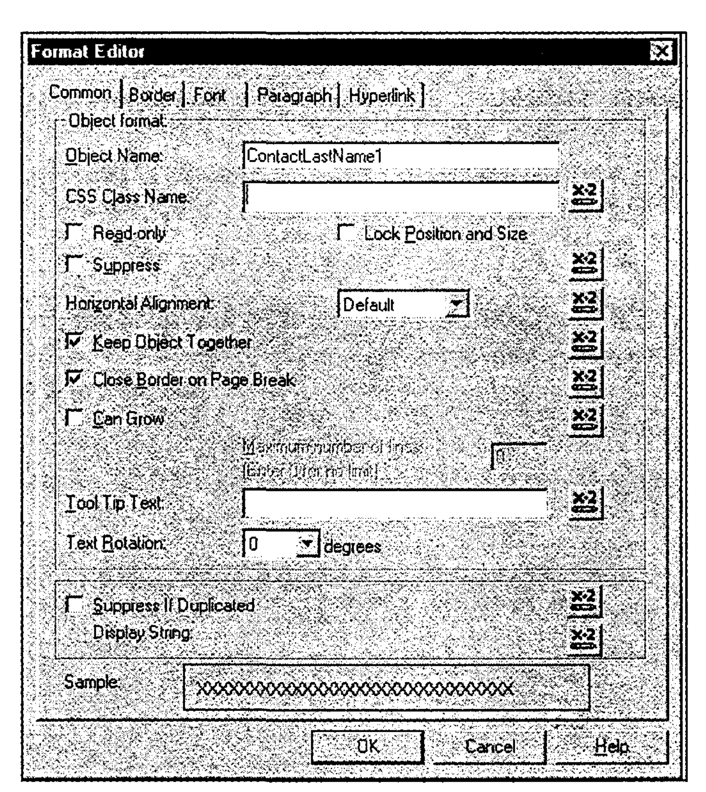


Рис. 2.15. Вкладка **Common** диалогового окна **Format Editor**

Вкладка **Common** диалогового окна **Format Editor** (рис. 2.15) содержит следующие опции форматирования:

- **Keep Object Together** — запрет на разрыв объекта при переходе на новую страницу. Если эта опция выбрана, то объект, не умещающийся на текущей странице, будет перенесен целиком на следующую страницу;
- **Close Border on Page Break** — если объект не умещается на текущей странице и переносится на следующую частично, т. е. разрывается на две части и каждая часть обрамляется рамками полностью;
- **Can Grow** — расположение текста объекта в несколько строчек;

- **Tool Tip Text** — создание ярлыка объекта, ярлык появляется в режиме просмотра;
- **Text Rotation** — вращение объекта, допускается горизонтальное или вертикальное размещение объекта;
- **Suppress if Duplicated** — если значение поля повторяется несколько раз, то показывается только первое значение, остальные скрываются.

Вкладка **Border** позволяет создать рамку для объекта. На вкладке **Font** можно установить размер, стиль и цвет шрифта.

Вкладка **Paragraph Formating** служит для форматирования текста, если он расположен в несколько строк.

С помощью вкладки **Hyperlink** (рис. 2.16) можно установить гиперссылку на другой объект текущего отчета, Web-сайт, адрес электронной почты или на другой отчет.

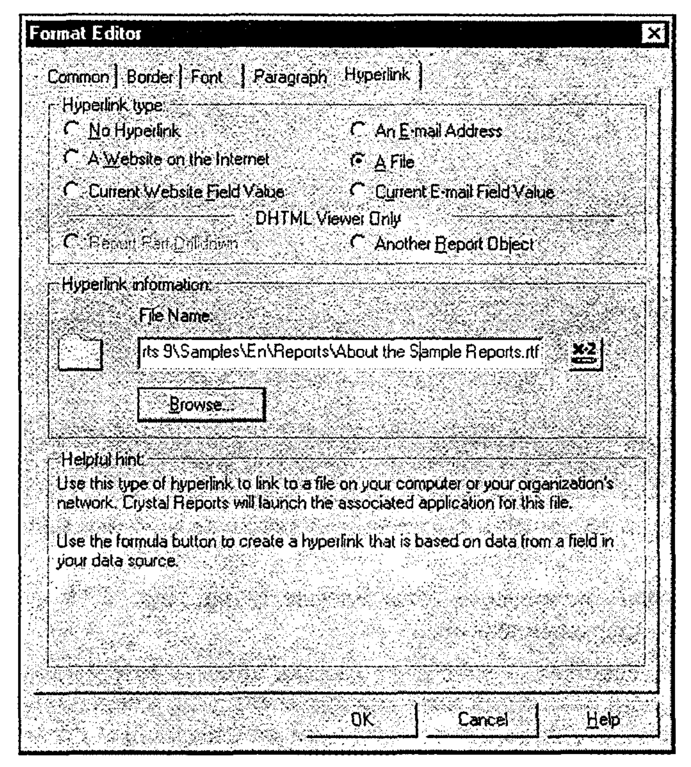


Рис. 2.16. Вкладка **Hyperlink** диалогового окна **Format Editor**

В группе радиокнопок **Hyperlink type** задается тип гиперссылки. В области **Hyperlink Information** для каждого типа гиперссылки следует внести допол-

нительную информацию. В табл. 2.2 приведены возможные типы гиперссылок, которые можно установить для объекта отчета в Crystal Reports 9.

Таблица 2.2. *Элементы управления панели инструментов диалогового окна **Field Explorer***

Тип гиперссылки	Описание	Содержимое раздела Hyperlink information
No Hyperlink	Отказ от гиперссылки для выбранного объекта, опция по умолчанию	Раздел недоступен
A Website on the Internet	Ссылка на адрес в Интернете. Адрес может быть статичным или генерироваться динамически с помощью формулы (кнопка справа от поля **Website Address**, подробнее о создании формул см. *гл. 5*), например: "http://www." + {Customer.Customer Name} + ".com"	Поле **Website Address**, в котором следует внести адрес Web-сайта (URL) Кнопка вызова редактора формул
Current Website Field Value	Ссылка на адрес в Интернете, хранящаяся в базе данных. Адрес должен храниться в выбранном поле	Раздел недоступен
An E-mail Address	Ссылка на адрес электронной почты. Адрес может быть статичным или генерироваться динамически с помощью формулы	Поле **E-mail Address**, в котором следует внести адрес электронной почты. Кнопка вызова редактора формул
A File	Ссылка на файл, хранящийся на локальном компьютере или в сети. Имя файла и путь могут быть статичными или генерироваться динамически с помощью формулы	Поле **File Name**, в котором следует внести имя файла и путь. Кнопка вызова редактора формул
Current E-mail Field Value	Ссылка на адрес электронной почты, хранящийся в базе данных. Адрес должен храниться в выбранном поле	Раздел недоступен
Another Report Object	Ссылка на объект текущего или какого-либо другого отчета. Если ссылка настроена на другой отчет, не текущий, он будет загружен в Crystal Reports 9 и выполнен	В поле **Select From** следует указать путь и имя файла отчета, на который создается ссылка, в поле **Object Name** — имя объекта отчета, в **Data Context** — значение поля, на которое будет установлен указатель, например: /Country[USA]/ Product Class[Bicycle]

Таблица 2.2 (окончание)

Тип гиперссылки	Описание	Содержимое раздела Hyperlink information
Report Part Drilldown	Выполняет ту же функцию, что и "высверливание" данных (Drill Down, см. *разд. 6.3*). Опция работает только в текущем отчете	Раскрывающийся список **Object Name**, в котором следует выбрать наименование объекта, на который будет установлен указатель

Упражнение 2.4

Увеличьте шрифт поля **Last Year's Sales** до 14 пунктов.

1. Откройте отчет, созданный в упражнении 2.3.
2. Щелкните в поле заголовка для его выделения.
3. Правой кнопкой мыши щелкните в поле и выберите **Format Field**.
4. В диалоге **Format Editor** выберите вкладку **Font** и установите размер 14.
5. Сохраните отчет.

Crystal Reports 9 позволяет включить в отчет изображения, хранящиеся в базе данных (рис. 2.17). Для этого нужно просто включить в отчет поле базы данных, в котором хранится изображение. Crystal Reports 9 автоматически распознает большой бинарный объект как изображение и поместит его в секции отчета.



В пакет Crystal Reports 9 входит отчет, иллюстрирующий возможность включения таких изображений. Его местонахождение: Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Reports\General Business\Employee Profile.rpt.

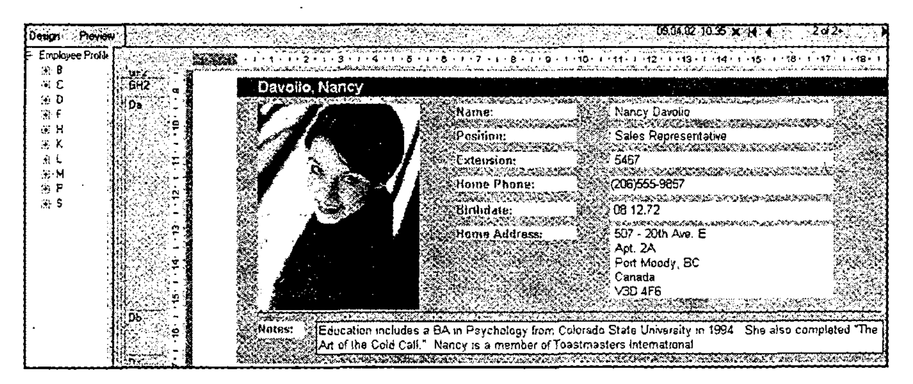


Рис. 2.17. Пример отчета, который содержит изображение, извлекаемое из базы данных

Упражнение 2.5

1. Используя в качестве источника данных таблицу **Employee**, создайте отчет и включите в него поля **First Name**, **Last Name**, **Photo**.
2. После выполнения данного упражнения вернитесь к отчету в упражнении 2.4.

2.3. Вставка в отчет текстовых объектов

Для вставки текстового объекта нужно щелкнуть на кнопке вставки текстового поля на панели инструментов (табл. 1.1) или выполнить команду меню **Insert | Text Object**. После этого следует щелкнуть на свободном месте в секции отчета, например, **Page Header**. Вставить текстовый объект можно как в режиме **Preview**, так и в **Design**.

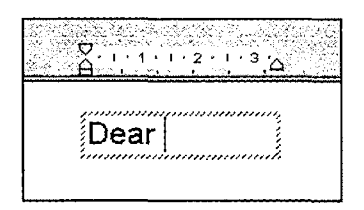


Рис. 2.18. Текстовый объект

После размещения текстового объекта Crystal Reports 9 переходит в режим редактирования. При помощи клавиатуры можно набрать текст, а в верхней части экрана появляется окно форматирования текстового объекта (рис. 2.18). Можно импортировать текст из текстового файла. Для этого в режиме редактирования следует щелкнуть правой кнопкой мыши на текстовом объекте и выбрать из контекстного меню **Insert From File**. Поддерживается импорт из файлов форматов ASCII, HTML и MS Word.

Текстовый объект в Crystal Reports 9 может содержать не только текст, но и поля базы данных, формулы, специальные поля и параметры (рис. 2.19). Чтобы внести в состав текстового объекта новое поле, нужно сначала создать его в какой-либо секции отчета, а затем, находясь в режиме редактирования, переместить его (Drag&Drop) внутрь текстового объекта.

Для того чтобы изменить свойства текстового объекта, необходимо выйти из режима редактирования (щелкнув на любом другом поле отчета) и щелкнуть на нем правой кнопкой мыши. В появившемся контекстном меню следует выбрать **Format Text**. Диалог **Format Editor**, служащий для редактирования свойств текстового объекта, полностью аналогичен соответствующему диалогу для редактирования свойств поля базы данных.

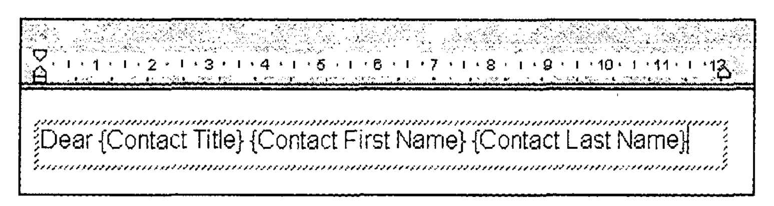


Рис. 2.19. Текстовый объект, содержащий поля базы данных

Для возвращения в режим редактирования следует дважды щелкнуть на текстовом объекте.

Если текстовый объект содержит поля базы данных, специальные поля или формулы, то в режиме редактирования можно задавать свойства для каждого из подобъектов отдельно.

Упражнение 2.6

1. Откройте отчет, созданный в упражнении 2.4.
2. Выполните команду меню **Insert | Text Object**.
3. Щелкните на свободном месте в секции **Page Header**.
4. Наберите "Отчет по продажам".
5. Установите для текстового объекта полужирный стиль и размер шрифта 14.
6. Сохраните отчет.

2.4. Вставка в отчет специальных полей

Помимо текстовых полей, в отчет могут быть включены специальные поля, содержащие дополнительную информацию, такую как номер страницы, номер записи, дата отчета и т. д. Для вставки специального поля необходимо в списке полей диалогового окна **Field Explorer** выбрать раздел **Special Field** (рис. 2.20).

Группа полей **Special Field** содержит следующие специальные поля.

- **Page Number** — номер страницы. Обычно размещается в секциях отчета **Page Header** или **Page Footer**.
- **Total Page Count** — общее количество страниц в отчете. Это поле может быть использовано в любой секции отчета. Например, **Total Page Count** может быть использовано для создания текстового объекта "Итого страниц в отчете — Y" или "страница X из Y", где X — **Page Number**, Y — **Total Page Count**. Включение этого поля в отчет может привести к увеличению времени формирования отчета, поскольку в этом случае Crystal Reports 9 при переходе на вкладку **Preview** формирует все страницы отчета.

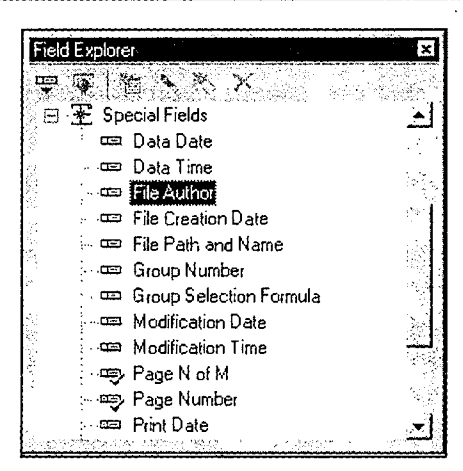


Рис. 2.20. Группа полей **Special Fields** в списке диалогового окна **Field Explorer**

- **Page N of M** — показывает номер текущей страницы из общего количества страниц в отчете. Например, если в режиме просмотра значение этого поля — "Page 2 of 8", то показана вторая страница из восьми, входящих в отчет.
- **Report Title** — заголовок отчета. Это поле может быть использовано в любой секции отчета. Конечно, в качестве заголовка отчета можно использовать и текстовый объект. Использование специального поля **Report Title** в качестве заголовка позволяет задать его значение из меню **File | Summary Info**. В результате значение **Report Title** будет свойством отчета и может быть использовано как параметр при вызове отчета из клиентского приложения, написанного на Delphi, MS Visual Basic или C++.
- **Report Comments** — комментарий к отчету. Использование этого поля аналогично использованию **Report Title**. Длина поля ограничена 256 символами.
- **File Path and Name** — имя файла отчета и путь к нему.
- **File Author** — имя автора отчета. Задается в диалоговом окне **Document Properties**, которое вызывается при помощи команды меню **File | Summary Info**.
- **File Creation Date** — дата создания шаблона отчета. По умолчанию присваивается системная дата компьютера, на котором создается отчет.
- **Print Date** — дата печати отчета. Может быть изменена командой меню **Report | Set Print Date | Time**.
- **Print Time** — время печати отчета. Может быть изменено командой меню **Report |.Set Print Date/Time**.
- **Data Date** — дата последнего обновления данных (refresh).

- **Data Time** — время последнего обновления данных (refresh).
- **Modification Date** — дата последнего изменения шаблона отчета.
- **Modification Time** — время последнего изменения шаблона отчета.
- **Record Number** — номер записи отчета. Может быть использован только в секции **Details**.
- **Group Number** — номер группы отчета. Может быть использован только в секциях **Group Header** или **Group Footer**.
- **Record Selection Formula** — поле отображает в режиме просмотра формулу, использующуюся для отбора записей. Более подробно создание формул и выборка записей будут рассмотрены в *главе 5*.
- **Group Selection Formula** — поле отображает в режиме просмотра формулу, использующуюся для отбора записей (фильтрация по агрегатным значениям).

2.5. Вставка в отчет параметров

Часто необходимо запускать отчет с минимальными изменениями. Для обеспечения таких изменений без редактирования отчета предназначены поля параметров. Значение параметра может быть задано пользователем перед выполнением отчета.

Использование параметра в отчете включает три стадии.

1. Создание поля параметра.
2. Включение параметра в отчет, например, в условие выборки или в формулу.
3. Задание параметра при выполнении отчета.

При создании параметра необходимо ввести его имя, тип, сопровождающий текст, набор значений и значение по умолчанию. Имя параметра в отчете будет показываться как **{?ParameterName}**. Допускаются следующие типы параметров:

- NumberVar — число;
- CurrencyVar — валюта;
- BooleanVar — логический (Boolean);
- DateVar — дата;
- DateTimeVar — дата и время (Timestamp);
- TimeVar — время;
- StringVar — строка.

Допускаются диапазоны значений параметров и множественные параметры. Для создания параметра необходимо в списке полей диалогового окна **Field**

Explorer переместить указатель на строку **Parameter Field** и щелкнуть кнопкой мыши на кнопке создания объекта (табл. 2.1). В появившемся диалоговом окне **Create Parameter Field** (рис. 2.21) необходимо задать имя **Name**, тип **Value type** и сопроводительный текст **Prompting text** параметра.

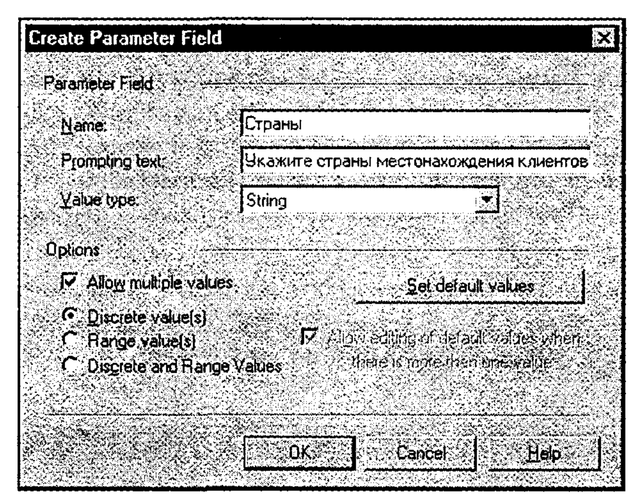


Рис. 2.21. Диалоговое окно **Create Parameter Field**

Рис. 2.22. Диалоговое окно **Set Default Values**

Для задания значений параметра следует щелкнуть на кнопке **Set default values**. Открывается диалоговое окно **Set Default Values** (рис. 2.22).

В качестве значения по умолчанию можно задать произвольное значение либо набор значений из базы данных. Тип значения по умолчанию, выбираемый из базы данных, должен соответствовать типу параметра. Если выбирается значение из базы данных, то в верхнем списке выбора диалогового окна **Set Default Values** следует выбрать таблицу, в которой хранятся эти значения. В списке выбора **Browse field** появляются поля таблицы, соответствующие типу параметра. После выбора поля в левом списке появляется перечень значений, содержащихся в поле. Щелкнув на кнопке >, можно внести значение в правый список, тем самым задав значения параметра по умолчанию.

При запуске отчета с параметрами появляется диалоговое окно **Enter Parameter Values** (рис. 2.23). В верхней части окна содержится список параметров, в нижней — список значений по умолчанию.

Рис. 2.23. Диалоговое окно **Enter Parameter Values**

Использование параметров для выборки и в формулах будет рассмотрено далее.

2.6. Вставка в отчет кумулятивных полей (running totals)

Кумулятивное поле (**running totals**) "накапливает" значение какого-либо поля базы данных от строки к строке отчета. Так, если значение поля суммы в первой строке — 1, во второй — 3, в третьей — 2, то значением кумулятивного поля будет 1 в первой строке, 4 (1+3) во второй и 6 (1+3+2) в третьей. Имя поля **running totals** в формулах начинается со знака "#", например, **{#my running total}**.

Для создания кумулятивного поля необходимо в списке полей диалогового окна **Field Explorer** переместить указатель на строку **Running Total Fields** и щелкнуть кнопкой мыши на кнопке создания объекта (табл. 2.1). Появляется диалоговое окно **Create Running Total Field** (рис. 2.24).

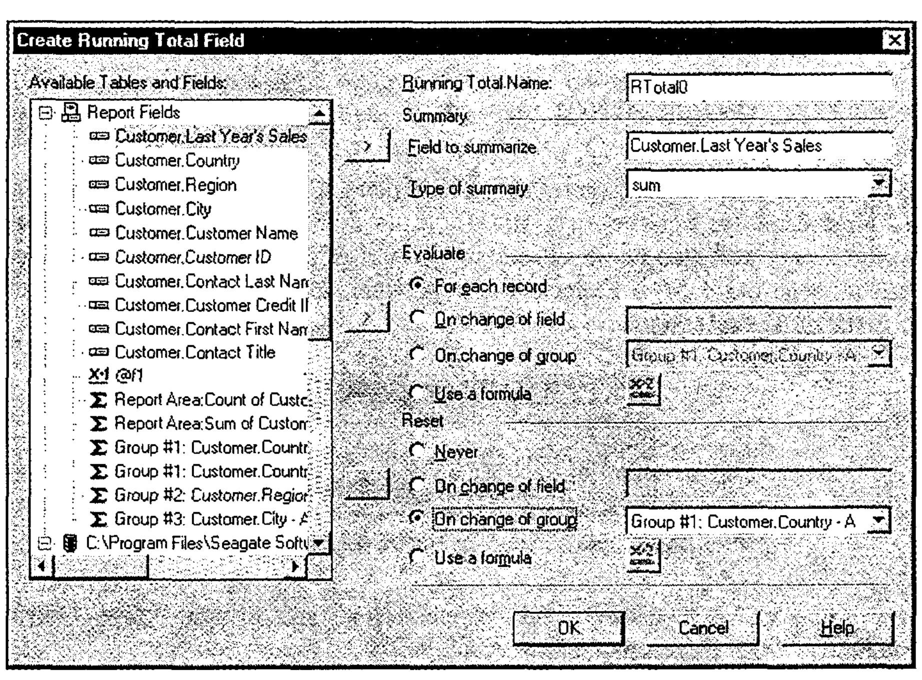


Рис. 2.24. Диалоговое окно **Create Running Total Field**

В правой верхней части диалогового окна расположено поле **Running Total Name**, в которое следует внести имя кумулятивного поля. В левой части диалога **Create Running Total Field** находится список допустимых полей, из которых можно выбрать то, значение которого будет суммироваться (**Field to summarize**). Для выбора поля необходимо в левом списке переместить указатель на строку с именем поля или колонки, щелкнуть кнопкой мыши

на кнопке >, расположенной слева от поля **Field to summarize**, и в раскрывающемся списке выбрать тип агрегатной функции (**Type of summary**).

Секция **Evaluate** позволяет определить порядок вычисления кумулятивного поля. Например, установка опции **For each record** задает изменение значения кумулятивного поля в каждой строке, при выборе опции **On change of group** значение кумулятивного поля будет изменяться только при переходе к следующей группе.

Секция **Reset** позволяет задать правила обнуления кумулятивного поля. Так, выбор опции **On change of group** означает, что для каждой новой группы кумулятивное поле будет вычисляться заново.

2.7. Вставка в отчет полей SQL expression

Вставка поля **SQL expression** доступна только при работе с SQL/ODBC-источниками. Имя поля **SQL expression** начинается со знака процента: **{%my SQL expression}**.

Для создания поля **SQL expression** необходимо в списке полей диалогового окна **Field Explorer** переместить указатель на строку **SQL expression Fields** и щелкнуть кнопкой мыши на кнопке создания объекта (табл. 2.1). Появляется диалоговое окно **SQL Expression Name** (рис. 2.25), в котором надо задать имя поля **SQL expression**.

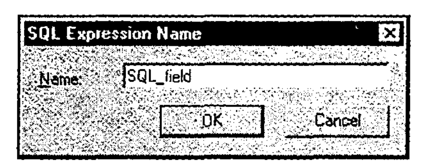


Рис. 2.25. Диалоговое окно **SQL Expression Name**

После щелчка по кнопке **OK** появляется диалоговое окно **Formula Workshop — SQL Expression Editor** (рис. 2.26).

Интерфейс диалогового окна **SQL Expression Editor** аналогичен интерфейсу **Formula Workshop — Formula Editor**, который будет рассмотрен в *главе 5*. Диалоговое окно содержит четыре списка в порядке слева направо.

1. Список формул.
2. Список полей таблиц баз данных.
3. Список функций.
4. Список операторов.

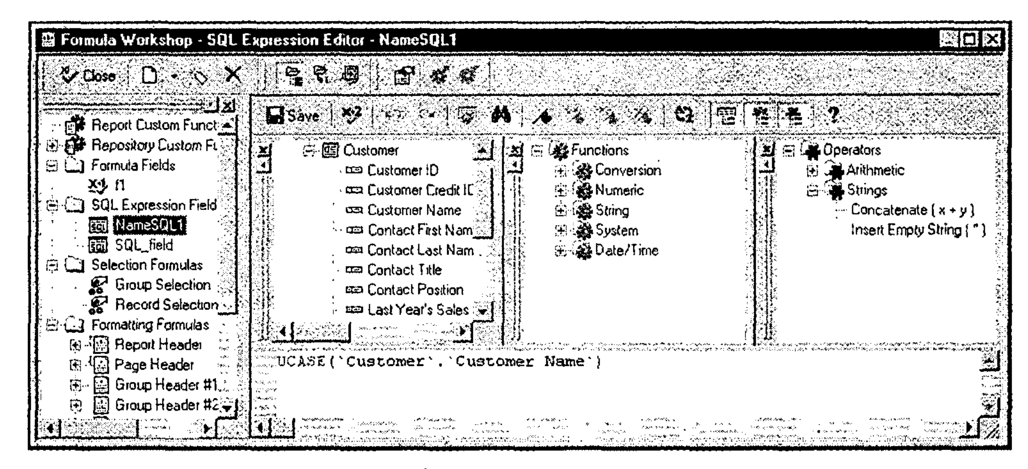


Рис. 2.26. Диалоговое окно
Formula Workshop - SQL Expression Editor

Для внесения элемента любого из четырех списков в текст формулы необходимо переместить указатель на соответствующую строку и дважды щелкнуть по нему кнопкой мыши.

В нижней части **SQL Expression** показывается окно с текстом формулы. Все окна и панель инструментов диалога перемещаемые — можно расположить их так, как удобно.

При обращении к реляционной базе данных Crystal Reports 9 генерирует SQL-выражение, в состав которого входит фрагмент, соответствующий полю **SQL expression**. Например, если значение поля **SQL Expression** есть **UCASE (`Customer`.`Customer Name`)** (см. рис. 2.26), то при обращении к серверу базы данных может быть сгенерировано выражение:

```
SELECT
 Customer.`Customer Name`, Customer.`Last Year's Sales`, Customer.`City`,
 {fn UCASE(Customer.`Customer Name`)}
FROM
 Customer.`Customer`
ORDER BY
 Customer.`City` ASC
```

Помимо параметров, полей базы данных, специальных, текстовых полей и полей **SQL expression**, отчет может включать формулы, которые будут рассмотрены в последующих главах.

2.8. Вставка линий и рамок

Отчету можно придать красивый вид, выделив данные линиями и рамками.

Для рисования линий можно использовать два метода:

1. Щелкнуть на кнопке "Нарисовать линию" в панели инструментов для вставки объектов в отчет (см. табл. 1.3).
2. Выполнить команду меню **Insert | Line**.

После этого курсор в окне отчета принимает вид карандаша. Для рисования линии надо передвинуть курсор на место начала линии и, нажав левую кнопку мыши, вести курсор к концу линии, затем отпустить кнопку.

Для рисования рамки тоже можно использовать два метода.

1. Щелкнуть на кнопке "Нарисовать прямоугольник" в панели инструментов для вставки объектов в отчет (см. табл. 1.3).
2. Выполнить команду меню **Insert | Box**.

В окне отчета появляется курсор в виде карандаша. Для рисования рамки следует передвинуть курсор на место левого верхнего угла рамки и, нажав левую кнопку мыши, вести курсор к правому нижнему углу рамки, затем отпустить кнопку.

Для форматирования линии или рамки следует переместить указатель на объект и щелкнуть правой кнопкой мыши. В контекстном меню следует выбрать пункт **Format Line** (форматирование линии) или **Format Box** (форматирование рамки). Появляется диалоговое окно **Format Editor** (рис. 2.27).

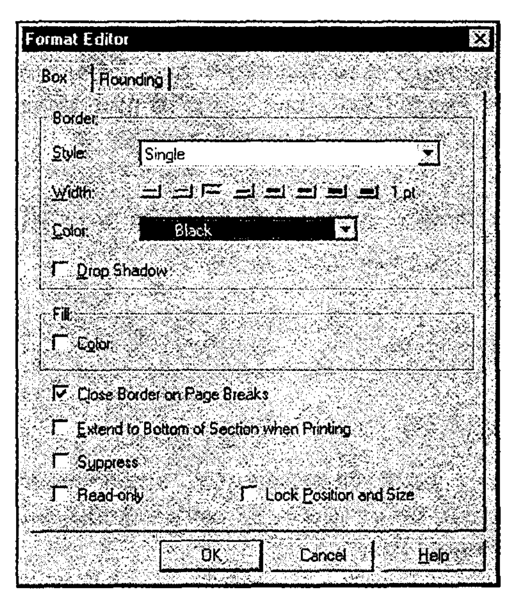


Рис. 2.27. Диалоговое окно для форматирования рамок и линий **Format Editor**

В диалоговом окне **Format Editor** можно изменить толщину, цвет или стиль линии и рамки. Вкладка **Rounding** (рис. 2.28) позволяет задать форму рамки — от прямоугольника до овала.

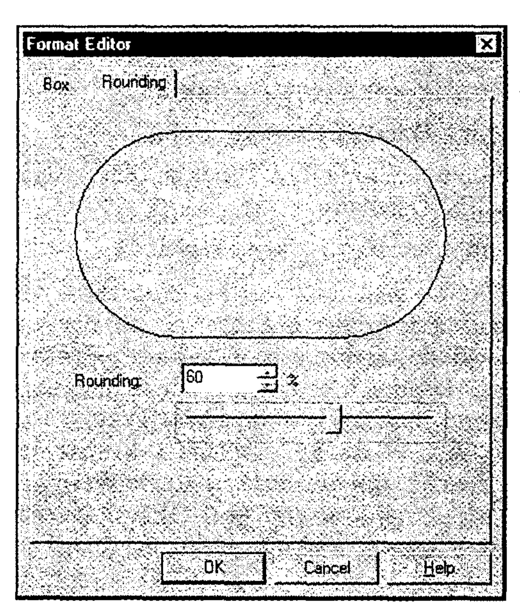


Рис. 2.28. Вкладка **Rounding** диалогового окна **Format Editor**

Рамки и линии, расположенные в секциях **Header** и **Footer**, будут показаны на каждой странице. Рамки и линии, расположенные в секции **Detail**, будут показаны на каждой записи.

Упражнение 2.7

1. Откройте отчет, созданный в упражнении 2.6.
2. Нарисуйте цветную линию под заголовком.
3. Нарисуйте цветную рамку вокруг наименования каждой группы.
4. Нарисуйте штриховую линию в секции **Page Footer**.
5. Сохраните отчет.

2.9. Вставка рисунка и OLE-объекта

Для создания красивых отчетов иногда целесообразно вставить в него рисунки. Например, это может быть логотип компании в заголовке. Для вставки рисунка необходимо:

- щелкнуть на кнопке "Вставка рисунка" в панели инструментов для вставки объектов в отчет (табл. 1.3) или выполнить команду меню **Insert | Picture**;

- в списке выбора **Тип файлов** диалогового окна **Открытие файла** выбрать какой-либо формат, например .BMP, GIF, PCX, TIFF, TGA;
- щелкнуть на **Открыть (ОК)** и расположить рисунок на отчете.

Упражнение 2.8

1. Откройте отчет, созданный в упражнении 2.7.
2. Щелкните на кнопке **Вставка рисунка**.
3. Выберите xtreme.bmp из каталога Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Databases.
4. Расположите рисунок в секции **Page Header**.

Вставка рисунка в отчет вышеописанным способом имеет один недостаток — рисунок невозможно редактировать, не выходя из отчета. Для редактирования такого рисунка необходимо изменить рисунок в другом приложении, а затем в Cristal Reports 9 удалить старый рисунок и вставить новый. Всех этих шагов можно избежать, используя технологию OLE (Object Linking and Embedding). OLE позволяет вставлять в отчет данные или образы, которые берутся из других приложений, и модифицировать их внутри отчета. Crystal Reports 9 является контейнером OLE и может возвращать объекты, которые созданы в других приложениях, если эти приложения поддерживают OLE.

Для вставки OLE-объекта следует выполнить команду меню **Insert | OLE object**.

Появляется диалоговое окно **Insert Object** (рис. 2.29).

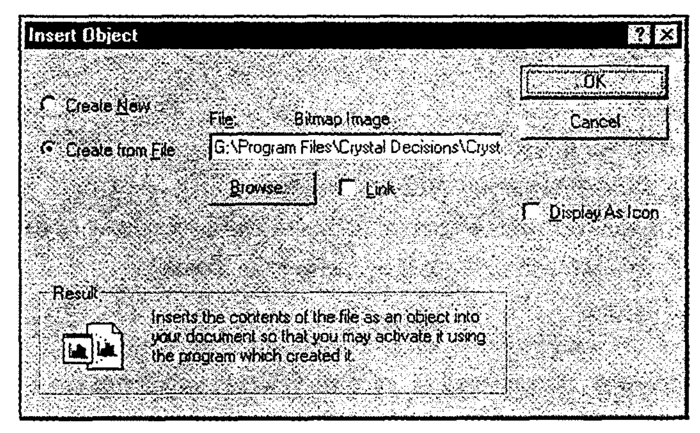


Рис. 2.29. Диалоговое окно **Insert Object**

В диалоговом окне **Insert Object** предлагается выбрать либо опцию **Create New** (создать новый объект), либо **Create from File** (выбрать существующий объект из файла).

При выборе опции **Create New** появится список **Object Type**. Если выбрать какое-либо приложение, открывается соответствующее окно, в котором можно создать новый объект.

Create from File означает вставку в отчет существующего объекта. Например, для внесения в отчет книги Microsoft Exel как объекта, нужно щелкнуть на **File** и выбрать имя файла из списка.

Рисунок и OLE-объект выглядят в отчете одинаково. Однако при выделении рисунка будет доступен лишь минимальный набор опций форматирования, а объект, например рисунок, можно модифицировать с помощью приложения, в котором объект был создан. Для редактирования объекта следует щелкнуть по нему дважды левой кнопкой мыши.

Упражнение 2.9

1. Откройте отчет, созданный в упражнении 2.7:
2. Вставьте xtreme.bmp как объект.
3. Отредактируйте рисунок, не выходя из Crystal Reports 9.

2.10. Использование хранилища объектов Crystal Repository

Хранилище объектов **Crystal Repository** является новым инструментом, появившимся в Crystal Reports версии 9. Оно позволяет централизованно хранить и использовать объекты отчета. Хранилище объектов может быть создано на локальном компьютере или на сервере, а объекты, единожды созданные и помещенные в хранилище, могут многократно использоваться одним или несколькими разработчиками для создания различных отчетов. Следовательно, хранилище объектов позволяет создавать единые корпоративные стандарты разработки отчетов. Кроме того, использование реляционных баз данных в качестве хранилища дает возможность эффективно решать проблемы безопасности данных. Отметим, что хранилище объектов входит во все варианты поставки Crystal Reports 9, кроме варианта Standard.

По умолчанию при инсталляции Crystal Reports 9 создается хранилище объектов с именем **Crystal Repository**. Это хранилище представляет собой базу данных MS Access и размещается в каталоге Program Files\Common Files\Crystal Decisions\2.0\bin\Repository.mdb. Для создания нового хранилища следует создать сначала базу данных, а затем с помощью стандартных средств операционной системы новый ODBC-источник данных. Вызов программы администрирования ODBC в системе Windows 2000 выполняется выбором иконки **Источники данных (ODBC)** окна **Администрирование**, которое, в свою очередь, находится в окне контрольной панели (**Control Panel**). Затем на вкладке **User DSN** диалогового окна **ODBC Data Source**

Administrator (рис. 2.30) щелкнуть на кнопке **Add** и с помощью мастера **Create New Data Source** задать свойства нового источника.

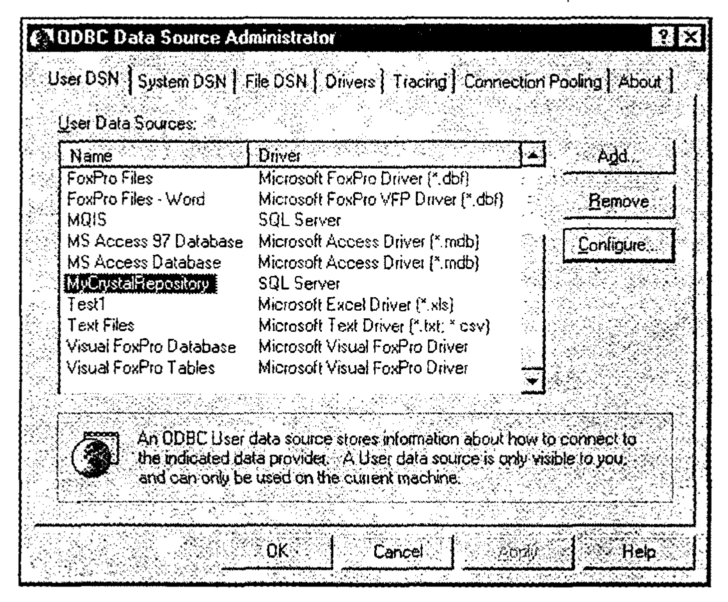


Рис. 2.30. Диалоговое окно **ODBC Data Source Administrator**

Следующим шагом является описание хранилища в файле orMap.ini. Этот файл по умолчанию находится в каталоге Program Files\Common Files\Crystal Decisions\2.0\bin. В файл orMap.ini должна быть добавлена строка в формате

<Наименование хранилища>=<наименование ODBC-источника>.

В имени хранилища нельзя использовать символы #, ", {, }, ; и /. Например, если создается новое хранилище New Repository, которое будет использовать ODBC-источник данных MyCrystalRepository, содержимое файла orMap.ini должно быть следующим:

```
п"ї# Syntax:
#
# <repository name>=<real data source name>
#
# Please note that <repository name> cannot contain these special
characters - # " { } ; /
# Any entry that contains such will be ignored.
# Also, this file is saved with UTF8 encoding. This allows <repository
name> to be of unicode characters.
#
New Repository=MyCrystalRepository
```

Crystal Reports 9 позволяет использовать только одно хранилище. Переконфигурация файла orMap.ini приводит к невозможности использования старого хранилища. Если во время создания нового источника данных и конфигурации файла orMap.ini Crystal Reports 9 был открыт, его следует закрыть и открыть вновь. Только после этого новое хранилище можно использовать.

Хранилище объектов может содержать несколько вложенных папок. Для создания новой папки следует перейти в диалоговое окно **Repository Explorer** (рис. 1.5), переместить указатель на строку древовидного списка, щелкнуть правой кнопкой мыши и выбрать в контекстном меню пункт **New Folder**. В появившейся строке списка следует внести имя новой папки.

Каждая папка может содержать объекты четырех типов:

1. Текстовые объекты;
2. Рисунки.
3. Специальные функции (Custom functions).
4. Запросы (commands или queries).

Объекты перечисленных типов могут быть добавлены в хранилище из отчета или извлечены из хранилища и помещены в отчет.

Для создания специальных функций предназначена среда разработки формул **Formula Workshop**, откуда они помещаются непосредственно в хранилище *(см. главу 5)*. Запросы (commands) могут быть созданы в диалоговом окне **Database Expert** и также помещены в хранилище. Для размещения запроса в хранилище необходимо при его создании в диалоговом окне **Add Command To Report** включить опцию **Add to Repository**.

Текстовый объект или рисунок может быть помещен в хранилище одним из двух способов:

- переместить текстовый объект или рисунок из рабочего окна отчета в диалоговое окно **Repository Explorer**;
- щелкнуть на текстовом объекте или рисунке правой кнопкой мыши и выбрать пункт меню **Add to Repository**.

В возникающем при этом диалоговом окне **Add Item** (рис. 2.31) следует указать наименование объекта — поле **Name**, автора — поле **Author** и ввести описание в поле **Description**. В нижней части окна содержится древовидный список папок хранилища **Location**. В нем необходимо выбрать папку, в которую будет помещен объект, после чего щелкнуть на кнопке **OK**.

Для внесения текстового объекта или рисунка из хранилища в отчет достаточно просто переместить его методом Drag&Drop из диалогового окна **Repository Explorer** в соответствующую секцию отчета.

Для внесения в отчет специальных функций следует воспользоваться диалоговым окном **Formula Expert**, которое будет описано в *главе 5*. Запросы,

размещенные в хранилище, являются источниками данных и могут быть включены в отчет наряду с прочими источниками данных с помощью диалогового окна **Database Expert**.

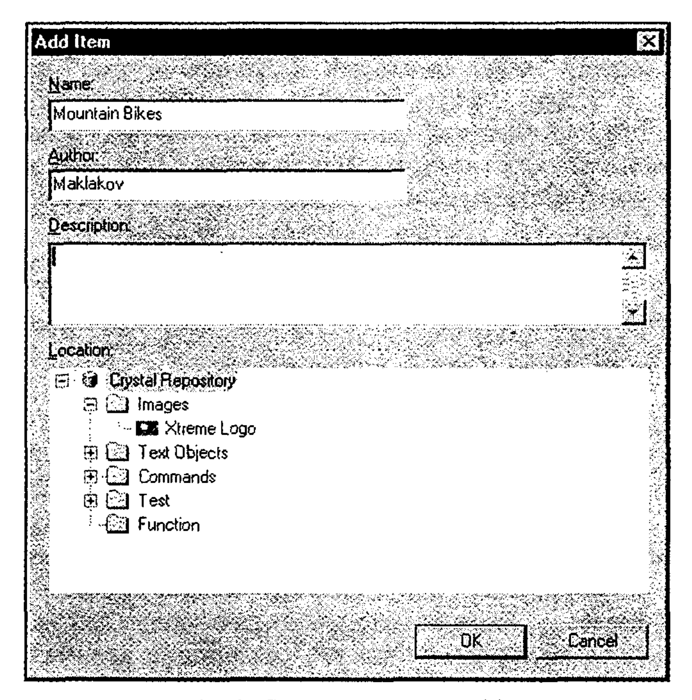


Рис. 2.31. Диалоговое окно **Add Item**

Перенесенные из хранилища в отчет рисунки и текстовые объекты нельзя редактировать, однако их можно обновить непосредственно в хранилище. При внесении в хранилище нового текстового объекта или рисунка на место старого возникает диалоговое окно **Add or Update Object** (рис. 2.32).

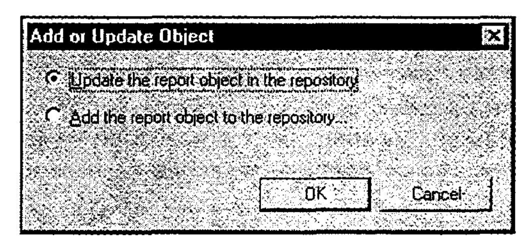
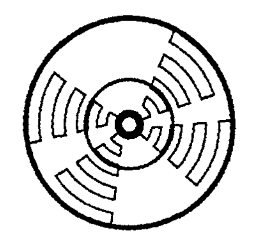


Рис. 2.32. Диалоговое окно **Add or Update Object**

Если выбрана опция **Update the report object in the repository**, то новый объект будет записан на месте старого.

Глава 3



Выборка, сортировка и группирование записей

3.1. Выборка записей

Обычно в отчете отображается только часть информации, хранящейся в базе данных. В *главе 2* было рассмотрено окно **Record Selection** мастера **Standard Report Creation Wizard**, позволяющее задать условия отбора записей при создании отчета. Рассмотрим, как можно ввести в существующий отчет дополнительные условия отбора.

Crystal Reports 9 допускает две возможности задания условий отбора записей.

1. Задание условий с помощью диалогового окна **Select Expert**.
2. Задание условий путем определения формул (Selection Formulas). Эта возможность будет рассмотрена в *главе 5*.

Диалоговое окно **Select Expert** (рис. 3.1) можно вызвать либо командой меню **Report | Select Expert**, либо щелкнув на кнопке выбора данных в панели инструментов для анализа (табл. 1.4).

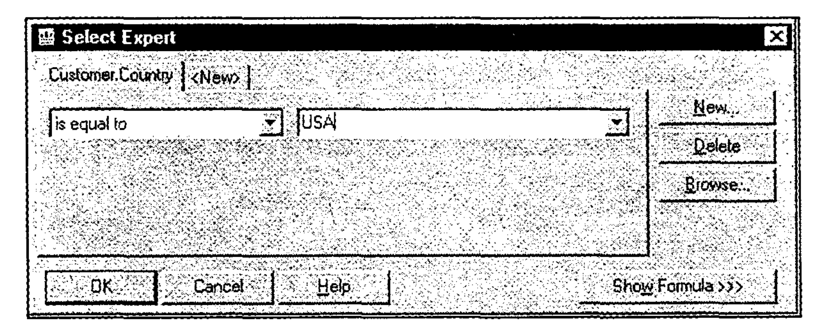


Рис. 3.1. Диалоговое окно **Select Expert**

Если в отчете не были заданы условия отбора, то сначала появляется диалоговое окно **Choose Field** (рис. 3.2) для выбора поля отчета или поля базы

данных, к которому будет применено условие отбора. Кнопка **Browse** в этом окне позволяет просмотреть первые 100 различных значений поля.

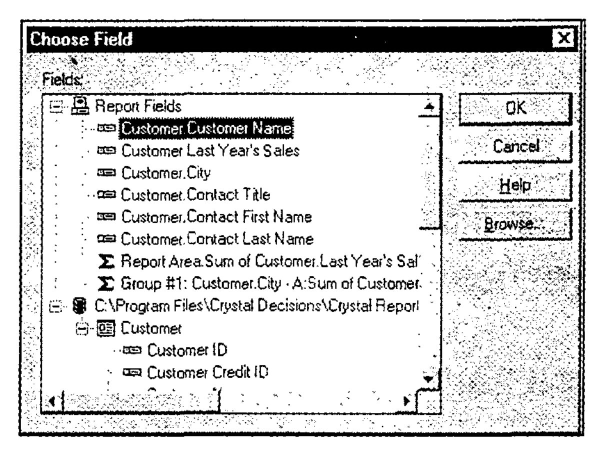


Рис. 3.2. Диалоговое окно **Choose Field**

Каждому логическому условию выборки соответствует одна вкладка в диалоговом окне **Select Expert**. Все условия объединяются логическим "и". Например, если установлено два условия:

```
{Customer.Last Year's Sales} is greater then $10000
```

и

```
{Customer.Country} is equal to France
```

то в отчет будет внесена информация только о клиентах, проживающих во Франции и одновременно сделавших заказы более чем на $10000.

Для удаления условия нужно перейти на соответствующую вкладку и щелкнуть на кнопке **Delete**. Для создания нового условия следует щелкнуть на кнопке **New**. Появляется диалоговое окно **Choose Field**, в котором следует выбрать поле и щелкнуть на кнопке **OK**. Автоматически создается новая вкладка в диалоговом окне **Select Expert** (рис. 3.3).

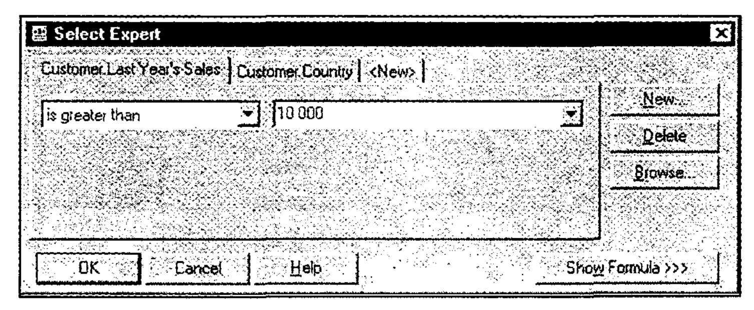


Рис. 3.3. Создание нового условия выборки в диалоговом окне **Select Expert**

Каждая вкладка диалогового окна в левой части содержит список выбора, предназначенный для задания логического оператора. Содержимое списка зависит от типа поля (числовое, строковое или дата). Служебное слово not используется для отрицания условия, например, условие is not equial включит в отчет строки, значения поля в которых не равно заданному.

Допускается использование следующих операторов.

- **equal to** — равенство. Применимо для поля любого типа.
- **one of** — равенство любому значению из заданного списка. Применимо для поля любого типа. В случае выбора оператора **one of** в правой части диалогового окна **Select Expert** появляется раскрывающийся список, содержащий значения поля и под ним список значений, на основе которого будет создано правило отбора. Для внесения значения поля в нижний список достаточно раскрыть верхний список, переместить указатель на строку с именем поля и щелкнуть по нему кнопкой мыши.
- **greater (less) then or equal to** — больше (меньше) или равно. Применимо для поля любого типа.
- **between** — задает верхнюю и нижнюю границу значений поля. Применимо для поля любого типа.
- **starts with** — выбирает текстовые поля, начинающиеся с заданного символа.
- **like** — выбор текстового поля по маске. Допускаются маски:
  - * — последовательность символов,
  - ? — один символ.
- **Formula** — установка выборки по формуле *(см. главу 5)*.
- **in the period** — применимо для поля типа дата и дата-время. Можно использовать формулы периода *(см. главу 5)*.

При попытке просмотра отчета после создания нового условия выборки выводится диалоговое окно **Change In Record Selection Formula Detected** (рис. 3.4), в котором предлагается использовать в отчете сохраненные данные или обновить их из источника данных щелчком на кнопке **Refresh Data**.

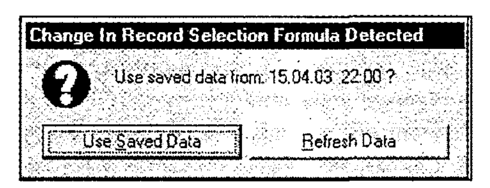


Рис. 3.4. Диалоговое окно
Change In Record Selection Formula Detected

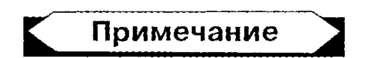
Используя возможность работы с сохраненными данными, можно, меняя правила отбора, последовательно включать в отчет все меньший и меньший набор данных. Однако в любое время можно принудительно обновить данные, выполнив команду меню **Report | Refresh Report Data** или нажав клавишу <F5>.

По умолчанию условие выборки учитывает регистр. Например, если по условию **Region is equial to CA**, выбираются записи с `CA`, а не `Ca`, `cA`, `ca`. Если в качестве источника данных был выбран ODBC, то можно настроить опцию чувствительность к регистру — **Database Server is Case-Insensitive**, выполнив команду меню **File | Options** на вкладке **Database** диалогового окна **Options**. В этом случае по условию **Region is equial to CA** будут выбираться все записи, содержащие и `CA`, и `Ca`, и `cA`, и `ca`.

Упражнение 3.1

Откройте отчет, созданный в упражнении 2.6.

1. С помощью диалогового окна **Select Expert** создайте дополнительное условие выборки для поля Region так, чтобы в отчет включались записи со значениями CA, NY и FL.
2. Просмотрите отчет. Убедитесь, что поле **Region** содержит только значения CA, NY или FL. Для этого внесите поле **Region** в секцию **Detail**.
3. Сохраните отчет.



Каталог Program Files/Crystal Decisions/Crystal Reports 9/Samples/En/Reports/ Feature Examples содержит ряд примеров, иллюстрирующих различные приемы построения отчетов. В частности, файлы Record Selection1.rpt и Record Selection2.rpt содержат отчеты, которые показывают возможность установки условий выборки с помощью окна **Select Expert**.

3.2. Группирование записей

По умолчанию записи в отчете располагаются в том порядке, в котором они располагаются в базе данных для настольных баз данных или в результирующем наборе данных для реляционных СУБД. Часто требуется сгруппировать записи, например, покупателей из одного города расположить вместе. Такая операция называется группированием записей — в одну группу включаются записи с одинаковым значением определенного поля. Сгруппировав записи, можно суммировать данные в каждой группе, например, подсчитать общую сумму продаж или количество покупателей в каждом городе. В *главе 2* было показано, как с помощью окна **Grouping** мастера **Standard Report Creation Wizard** добавить группу в новый отчет.

Вставить группу в уже существующий отчет можно двумя способами:

1. С помощью диалогового окна **Group Expert**.
2. С помощью диалогового окна **Insert Group**.

Диалоговое окно **Group Expert** (рис. 3.5) можно вызвать, выполнив команду меню **Report | Group Expert** или щелкнув на соответствующей кнопке в панели инструментов для анализа (табл. 1.4).

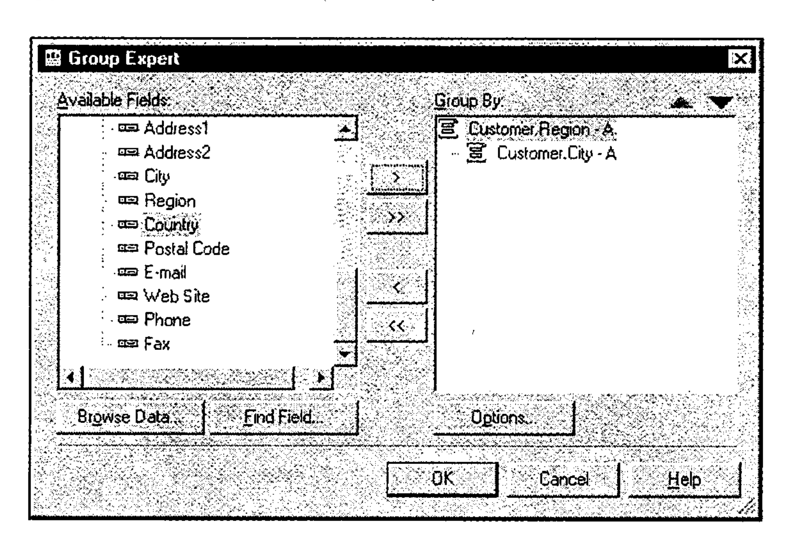


Рис. 3.5. Диалоговое окно **Group Expert**

Левый список окна — **Available Fields** содержит поля отчета и поля таблиц, на основе которых могут быть созданы группы. Правый список **Group by** содержит группы отчета. Для создания в отчете новой группы необходимо в левом списке переместить указатель на строку с наименованием поля, по которому будет проведено группирование, и щелкнуть кнопкой мыши на кнопке >. Над списком **Group by** расположены кнопки, позволяющие изменять порядок группировки.

Для вызова диалогового окна **Insert Group** (рис. 3.6) следует выполнить команду меню **Insert | Group**.

Для вставки группы необходимо в верхнем списке вкладки **Common** диалогового окна **Insert Group** выбрать поле для группирования и порядок, в котором группы должны показываться. Например, для рассматриваемого отчета можно выбрать группирование по полю **Region**. Можно сгруппировать информацию по полям отчета или даже по полям, которые не входят в отчет.

Порядок сортировки групп можно установить в нижнем списке выбора. Группы могут располагаться в порядке возрастания — **in ascending order**

(от A до Z и от 1 до 9) и в порядке убывания — **in descending order** (от Z до A и от 9 до 1). При выборе **in original order** сортировка групп не производится. Наиболее интересна сортировка **in specified order**. Она позволяет установить группирование по признаку, который не хранится в базе данных. При выборе сортировки **in specified order** в диалоговом окне **Insert Group** появляется вкладка **Specified Order** (рис. 3.7).

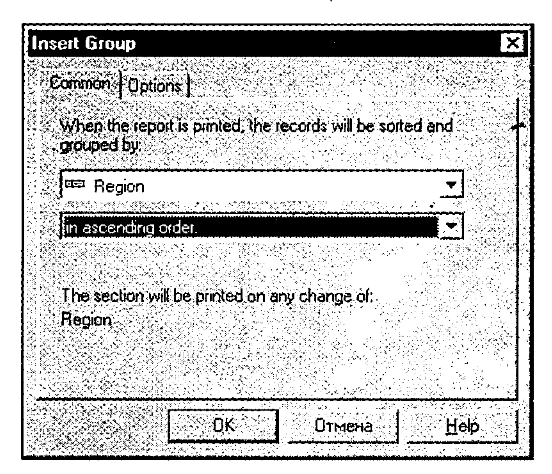


Рис. 3.6. Диалоговое окно **Insert Group**

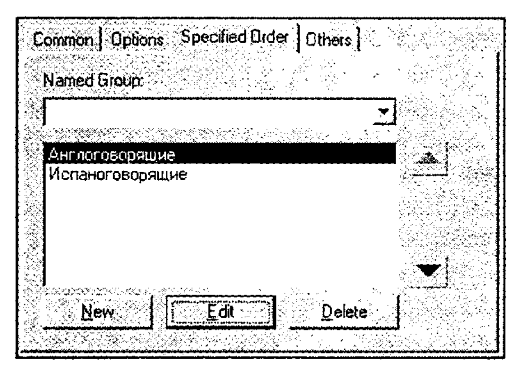


Рис. 3.7. Вкладка **Specified Order** диалогового окна **Insert Group**

Например, необходимо выделить группы англоговорящих или испаноговорящих стран. Для этого сначала нужно описать эти группы. Если щелкнуть на кнопке **New**, появляется диалог **Define Named Group** (рис. 3.8).

В этом диалоге можно задать логическое условие, описывающее группу. На рис. 3.8 испаноговорящие страны — Аргентина, Мексика и Испания.

Вкладка **Options** диалогового окна **Insert Group** служит для настройки свойств групп.

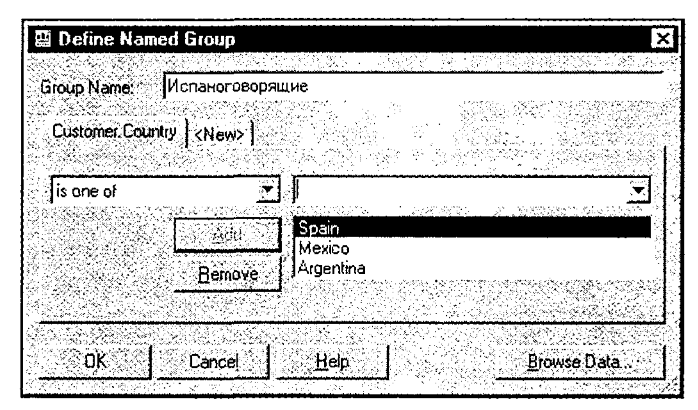


Рис. 3.8. Диалоговое окно **Define Named Group**

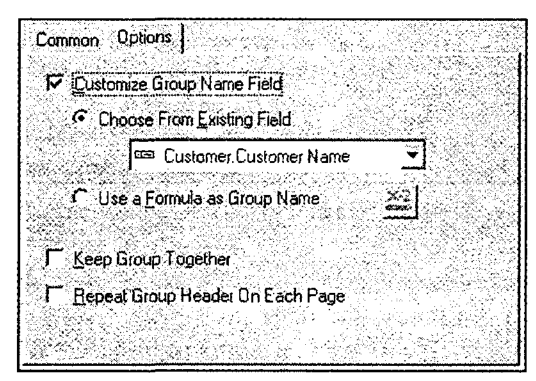


Рис. 3.9. Вкладка **Options** диалогового окна **Insert Group**

Установка опции **Keep group together** предотвращает разрыв группы на разные страницы.

Использование опции **Repeat Group Header On Each Page** позволяет повторить заголовок группы на каждой странице, если группа располагается на разных страницах.

При создании каждой группы в отчет добавляются новые секции — **Group Header** и **Group Footer**. По умолчанию в секцию **Group Header** добавляется заголовок группы — значение поля, по которому производится группирование. Например, если производится группирование по городу, в качестве названия группы берется название города. Если группирование выполняется по числовому полю или дате, в качестве названия группы берется конвертированное в строку значение поля. Иногда не вполне удобно брать в качестве названия группы значение поля. Так, если группирование производится по номеру клиента, логичнее было бы в качестве названия взять имя клиента, а не его номер. Опция **Customize Group Name Field** (рис. 3.9) позволяет

в качестве названия группы указать значение любого поля или формулы, например:

```
ToText ({Customer.CustomerID}) + " : " + {Customer.CustomerName})
```

Новая группа автоматически становится внутренней. Если в отчете уже существовала группа, то необходимо следить за правильностью логики группирования. В рассматриваемом нами примере в одной стране может быть несколько городов, а не наоборот. Изменить порядок групп несложно. Для этого, находясь на вкладке **Design**, нужно переместить методом Drag&Drop заголовки секций групп (рис. 3.10).

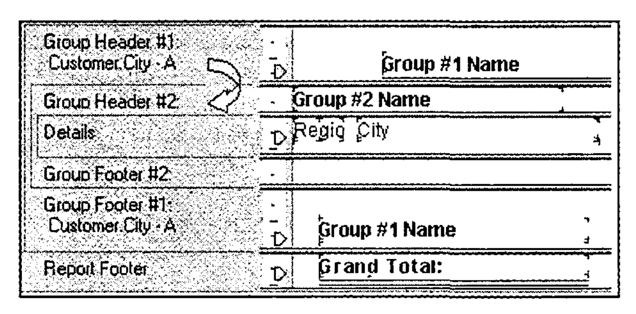


Рис. 3.10. Изменение порядка группирования

Если группирование производится по полю типа дата, на вкладке **Common** диалога **Insert Group** появляется список выбора, позволяющий задать опции группировки (рис. 3.11). Например, в одну группу могут быть объединены записи по значению поля типа дата за один день, одну неделю, месяц, квартал, год и т. д.

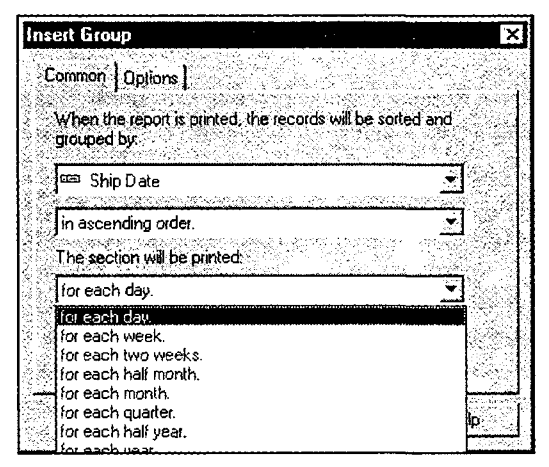


Рис. 3.11. Группирование по полю типа дата

Упражнение 3.2

1. Откройте отчет, созданный в упражнении 2.6.
2. В диалоговом окне **Select Expert** удалите все условия выборки.
3. Сгруппируйте отчет по полю **Country** и установите сортировку **in specified order**.
4. Создайте группы англоязычных и испаноязычных стран, как описано ранее.
5. Выделите заголовок группы **Country** фоном или цветом.
6. Сделайте группу по странам внешней по отношению к группе по городам.
7. Сохраните отчет.

Crystal Reports 9 позволяет производить группирование на основе данных, образующих иерархическую рекурсию. Иерархическая рекурсия — это структура данных, в которой таблица ссылается на саму себя. Пример такой таблицы в обозначениях стандарта IDEF1X показан на рис. 3.12. Например, в таблице **Employee** хранится информация о сотрудниках фирмы. Поле **Employee ID** содержит номер сотрудника, а поле **Reports To** — номер руководителя сотрудника, при этом информация о нем хранится в той же таблице. Поле **Reports To** может принимать значение NULL — это показывает, что у сотрудника может и не быть руководителя.

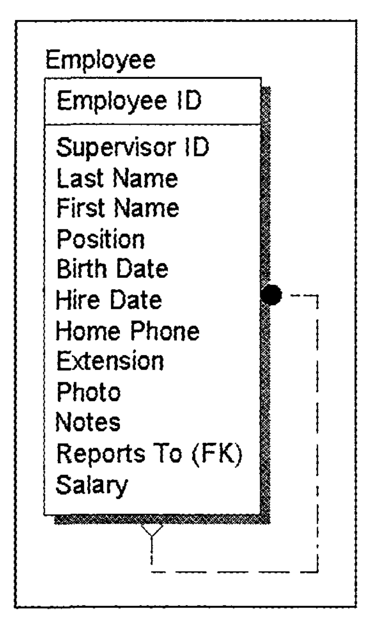


Рис. 3.12. Пример таблицы с иерархической рекурсией

Crystal Reports 9 позволяет создавать древовидный отчет, раскрывающий структуру подчинения сотрудников организации. Для этого необходимо создать группу по первичному ключу таблицы **Employee ID**, затем выполнить команду меню **Report | Hierarchical Grouping Options**. В открывшемся диалоговом окне **Hierarchical Options** (рис. 3.13) следует включить опцию **Sort Data Hierarchically** и указать родительское поле группы **Parent ID Field**, в примере это — **Reports To**.

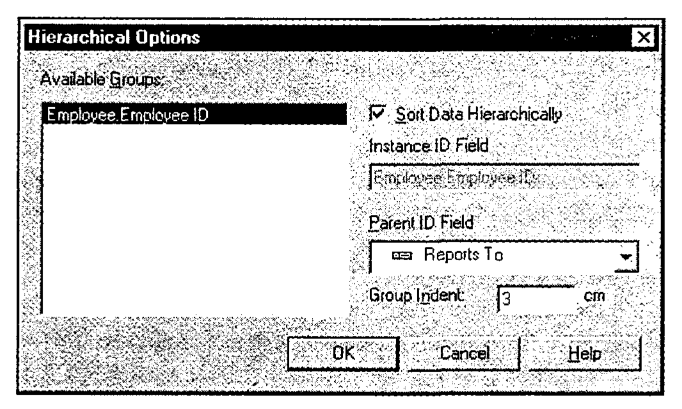


Рис. 3.13. Диалоговое окно **Hierarchical Options**

В поле **Group Indent** указывается смещение вправо группы нижнего уровня отчета в сантиметрах. Уровень вложений не ограничен. Пример отчета с иерархической группировкой показан на рис. 3.14.

Employee ID	Last Name	First Name
2	Fuller	Andrew
1	Davolio	Nancy
3	Leverling	Janet
11	Smith	Tim
12	Patterson	Caroline
4	Peacock	Margaret
5	Buchanan	Steven
6	Suyama	Michael
7	King	Robert
9	Dodsworth	Anne
8	Callahan	Laura
10	Hellstern	Albert
13	Brid	Justin
14	Martin	Xavier
15	Pereira	Laurent

Рис. 3.14. Пример отчета с иерархической группировкой

3.3. Сортировка записей

После группировки информации в отчете, внутри групп она располагается в произвольном порядке. Для упорядочения записей необходимо установить порядок их сортировки, например по алфавиту.

Для установки порядка сортировки необходимо выполнить команду меню **Report | Record Sort Expert** или щелкнуть на соответствующей кнопке панели инструментов для анализа (табл. 1.4). Появляется диалоговое окно **Record Sort Order** (рис. 3.15).

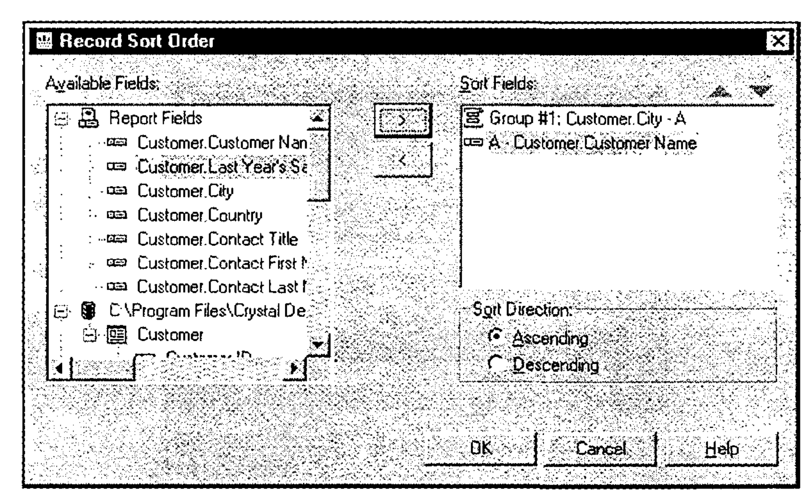


Рис. 3.15. Диалоговое окно **Record Sort Order**

В диалоговом окне **Record Sort Order** по умолчанию устанавливается сортировка по тем полям, по которым установлено группирование. Для задания дополнительной сортировки внутри групп необходимо переместить указатель на строку с наименованием поля в левом иерархическом списке и щелкнуть по кнопке >. Поле перемещается в правый список. Опции **Ascending** и **Descending** определяют порядок сортировки по возрастанию и убыванию соответственно.

Упражнение 3.3

1. Откройте отчет, созданный в упражнении 3.1.
2. Установите сортировку информации по полю **Customer Name** в порядке возрастания.
3. Сохраните отчет.

3 Зак. 1021

3.4. Вставка агрегатных полей (Summary)

После группировки информации в отчете можно добавить в него некоторую суммирующую информацию, например, количество компаний в каждом городе.

Для отображения суммирующей информации служат агрегатные (**Summary**) поля, такие как сумма, максимальное, минимальное или среднее значение. По умолчанию они размещаются в секции отчета **Group Footer**.

Для вставки поля **Summary** следует переместить указатель на поле в секции **Detail** и щелкнуть по нему правой кнопкой мыши. В контекстном меню следует выполнить команду **Insert | Summary**. Появляется диалоговое окно **Insert Summary** (рис. 3.16).

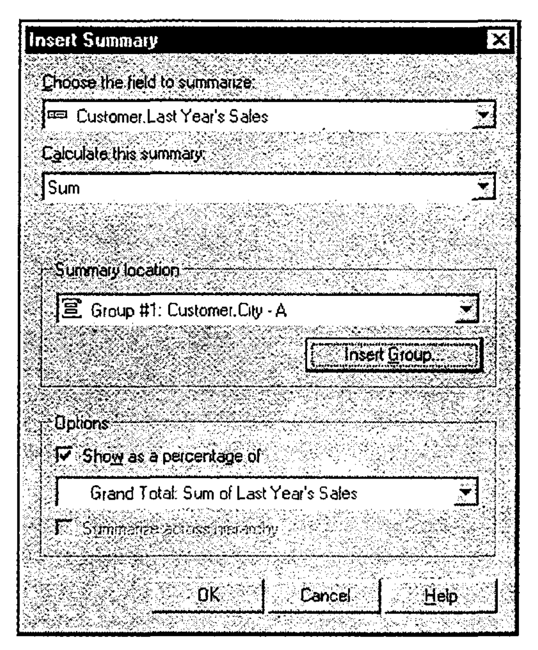


Рис. 3.16. Диалоговое окно **Insert Summary**

В раскрывающемся списке **Choose the field to summarize** в верхней части диалогового окна **Insert Summary** следует выбрать поле, по которому будет производиться вычисление, а в списке **Calculate this summary** — тип агрегатной функции.

Раскрывающийся список **Summary location** служит для выбора группы, в рамках которой будет производиться вычисление. Результат вычисления — агрегатное поле — будет помещен в секцию **Group Footer** выбранной группы. Если выбрать значение **Grand Total (Report Footer)**, то вычисление будет производиться по всему отчету, а результат будет помещен в секцию **Report Footer**.

Если группы, по которой необходимо произвести вычисление, не существует, можно щелкнуть на кнопке **Insert Group**. Возникает диалоговое окно **Insert Group** (рис. 3.6), в котором можно создать новую группу.

Опция **Show as a percentage of** в нижней части диалогового окна **Insert Summary** позволяет показать вычисленное значение как процент от другого агрегатного поля, которое указывается в нижнем списке выбора. Например, если в списке **Summary location** указано значение **Group #1: Customer.City-A**, а в нижнем списке — **Grand Total: Sum of Last Year's Sales** (рис. 3.16), то в отчете будет показываться не абсолютное значение продаж в каждом городе, а процентное отношение к общему количеству продаж.

Для вставки агрегатного поля в отчет следует щелкнуть на кнопке **OK**.

Упражнение 3.4

1. Откройте отчет, созданный в упражнении 3.3.
2. Подсчитайте количество клиентов в каждом городе, для этого вставьте **Subtotal (count)** для поля **Customer Name**.
3. Вставьте текстовый объект в качестве подписи.
4. Сохраните отчет.

Упражнение 3.5

1. Откройте отчет, созданный в упражнении 3.4.
2. Подсчитайте общее количество клиентов в отчете, для этого вставьте **Grand Total (count)** для поля **Customer Name**.
3. Вставьте текстовый объект в качестве подписи.
4. Сделайте так, чтобы обе подписи выделялись шрифтом или цветом.
5. Сохраните отчет.

3.5. Порядок сортировки группы по суммирующим значениям. Сортировка TopN

В *разд. 3.2* было показано, как группировать записи и сортировать группы в порядке убывания или возрастания заголовков групп. Однако группы могут быть отсортированы и по результату вычислений суммирующих значений. Например, может быть установлена такая группировка, когда города с наибольшим количеством клиентов будут расположены в верхней части отчета. Для установки такой сортировки следует выбрать пункт **Group Sort Expert** из меню **Report**. Появится диалог **Group Sort Expert** с вкладкой для каждого агрегатного поля отчета (рис. 3.17). Заголовок каждой вкладки — это имя поля,

по которому производилось группирование. Например, если в отчете производится подсчет количества клиентов в каждой стране и в каждом городе, группы будут показываться на вкладках **Customer.City** и **Customer.Country**.

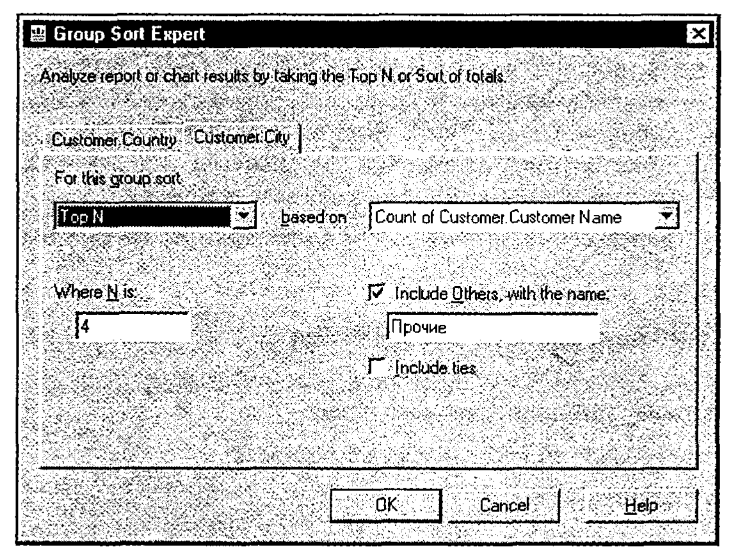


Рис. 3.17. Диалоговое окно **Group Sort Expert**

Опции сортировки групп, которые можно выбрать в раскрывающемся списке **For this group sort** диалогового окна **Group Sort Expert** приведены в табл. 3.1.

Таблица 3.1. *Опции сортировки групп*

Опция сортировки	Описание опции
Sort all	Будут отсортированы и напечатаны все группы
Top N	Будут отсортированы и напечатаны группы с наибольшим значением агрегатного поля, например города, где количество клиентов больше, чем в остальных. В поле **Where N is** нужно указать, сколько групп — N — нужно включать в отчет
Botton N	Будут отсортированы и напечатаны группы с наименьшим значением агрегатного поля. В поле **Where N is** нужно указать, сколько групп — N — нужно включать в отчет
Top Percentage	Будут отсортированы и напечатаны группы со значением агрегатного поля, большим указанного количества в процентах от общего количества. Например города, где количество клиентов больше, чем 20 % от общего количества клиентов в отчете. В поле **Where Percentage is** нужно указать величину процентов

Таблица 3.1 (окончание)

Опция сортировки	Описание опции
Botton Percentage	Будут отсортированы и напечатаны группы со значением агрегатного поля, меньшим указанного количества в процентах от общего количества. В поле **Where Percentage is** нужно указать величину процентов

Для сортировки всех групп нужно выбрать **Sort all** и затем суммирующее поле, по которому производится сортировка. Например, если одновременно подсчитывается количество клиентов и общая сумма заказов за прошлый год, в списке будут присутствовать значения **Count of Customer.Customer Name** и **Sum of Last Year's Sales**. Если выбрать **Count of Customer.Customer Name** и **ascending**, в отчете будут представлены сначала города с наименьшим, затем с наибольшим количеством клиентов, т. е. упорядоченные по возрастанию. Если выбрать **Sum of Sum of Last Year's Sales** и **ascending**, в отчете будут представлены сначала города с наименьшими, затем с наибольшими суммами продаж.

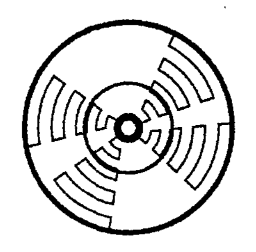
Опция **Top N** предоставляет мощные средства сортировки. Если нужно вынести наверх несколько групп с наибольшим значением суммирующего поля, можно использовать опцию **Top N** и указать N, например, 3, 4, 5.

Опция **Include Others, wish the name** указывает, будут ли включены в отчет записи, не относящиеся к отсортированным группам. Если опция включена, остальные записи будут объединены и включены в группу "прочие". Заголовок для группы "прочие" можно изменить, по умолчанию он — Other.

Упражнение 3.6

1. Откройте отчет, созданный в упражнении 3.5.
2. Упорядочьте группы по городам так, чтобы три города с наибольшим количеством клиентов были представлены в верхней части группы по странам, а остальные города — в группе "прочие".
3. Сохраните отчет.

Глава 4



Диаграммы и географические карты

4.1. Вставка диаграммы

Очевидно, что представление информации в графическом виде гораздо нагляднее, чем просто в виде таблиц. Crystal Reports 9 позволяет создавать диаграммы с использованием специального инструмента — **Chart expert**. Опытные пользователи могут улучшить изображение диаграммы с помощью более изощренного и мощного редактора. Рассмотрим сначала создание диаграммы с помощью **Chart expert**.

Для вставки диаграммы в отчет необходимо выполнить команду меню **Insert | Chart** или щелкнуть на кнопке вставки диаграммы панели инструментов для вставки объектов в отчет (табл. 1.3).

Появляется мастер отчетов **Chart Expert**.

Chart Expert — это диалоговое окно с вкладками. Первая вкладка — **Type** (рис. 4.1) позволяет выбрать один из 14 предопределенных типов диаграмм. После выбора типа диаграммы можно использовать прочие вкладки и дополнительные опции выбора данных, размещения текста к диаграмме и т. д. Рассмотрим каждую вкладку подробно.

Для выбора типа на вкладке **Type** диаграммы следует щелкнуть на соответствующей строке в левом списке. После этого в правой части вкладки показываются подтипы выбранного типа диаграммы. Для выбора подтипа необходимо щелкнуть на соответствующей кнопке. Рассмотрим основные типы диаграмм.

- **Bar** — показывает значение функции в виде полосок. На такой диаграмме хорошо видны относительные значения каждой величины. Диаграмма типа **Bar** имеет шесть подтипов, в том числе: **Side by side bar chart** и **Stacked bar chart**. Диаграмма **Side by side bar chart** отображает данные как серию вертикальных полосок. Этот тип рекомендуется для отображения значений функции одного аргумента, например таких, как объем продаж в различных городах. Диаграмма **Stacked bar chart** отображает данные как

серию разделенных вертикальных полосок. Такая диаграмма будет наглядно отображать объем продаж в различных городах за последние годы. Полоска, соответствующая каждому городу, будет разделена на несколько частей.

- **Line** — представляет данные в виде нескольких точек, соединенных линией. Такая диаграмма хорошо подходит для отображения данных с очень большим количеством числовых значений.
- **Area** — изображает данные в виде цветных областей. Эта диаграмма подходит, когда важно отобразить как абсолютное значение, так и процентное соотношение.
- **Pie** — изображает данные в виде круга, разделенного на сегменты. Такая диаграмма удобна для отображения относительной доли какой-либо величины. Подтип **Multiple Pie** изображает данные в виде нескольких кругов, разделенных на сегменты.
- **Doughnut** — подобен типу **Pie**, но изображает данные не в виде круга, а в виде кольца. Имеет подтип **Multiple Pie Doughnut** — нескольких колец, разделенных на сегменты. Если нужно отобразить процентное соотношение величин, например, долю продаж в различных городах в зависимости от суммарного объема продаж, то удобно использовать тип **Pie** или **Doughnut**.

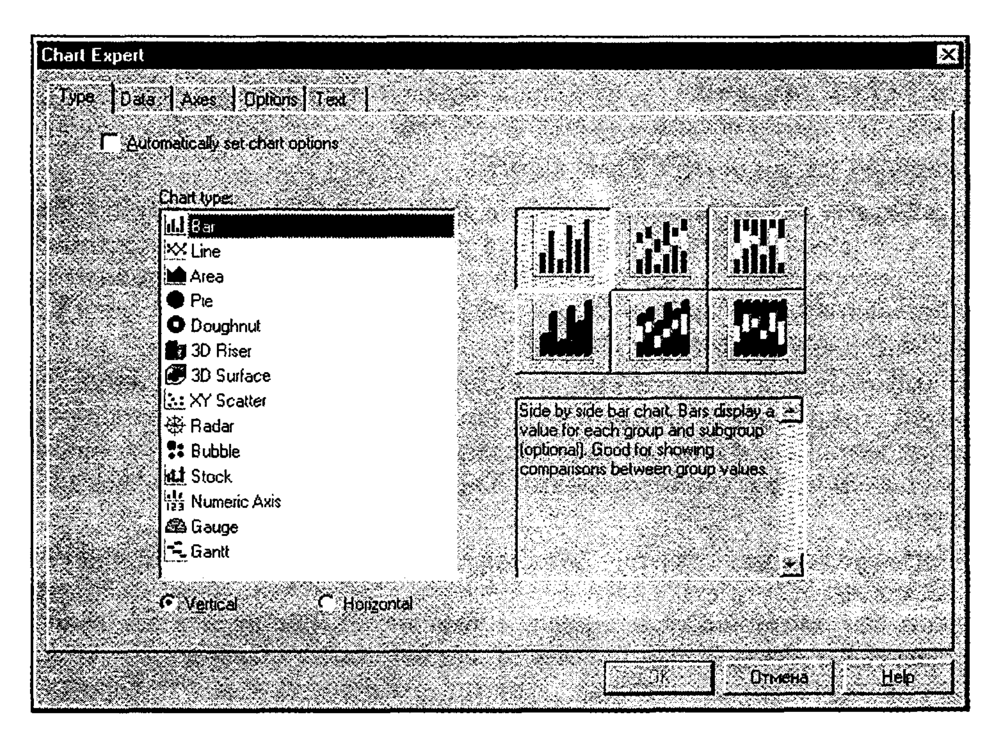


Рис. 4.1. Вкладка **Type** диалогового окна **Chart Expert**

- **3-D Riser** — изображает данные в виде трехмерных объектов. Удобна для отображения функции двух аргументов, например, если необходимо рассмотреть объем продаж различных товаров в разных городах.
- **3-D Surface** — представляет данные в виде трехмерной поверхности.
- **XY Scatter** — представляет данные в виде символов на координатной плоскости XY.
- **Radar** — представляет данные в виде символов в радиальных координатах. Чем больше величина, тем дальше располагается символ от центра.
- **Bubble** — представляет данные в виде символов различной величины на координатной плоскости XY. Диаграмма вида **Bubble** похожа на **XY Scatter**, но более удобна для отображения функции двух переменных.
- **Stock** — показывает диапазон изменения данных относительно максимального и минимального значения.
- **Numeric Axis** — отличается от диаграмм **Line** и **Bar** тем, что по оси абсцисс откладываются числовые значения, а не номера строк.
- **Gauge** — удобна, когда нужно наглядно представить небольшой набор значений. Значения откладываются на секторах "циферблата", причем на "циферблате" можно задать области различного цвета (зеленую, желтую, красную).
- **Gantt** — диаграмма Гантта, предназначена для отображения продолжительности работ. Каждая запись в отчете, в который вставляется диаграмма Гантта, должна иметь поля с датой (временем) начала и окончания работы.

Вкладка **Data** (рис. 4.2) позволяет определить, данные из каких полей отчета будут использованы при построении диаграммы. Кроме того, на этой вкладке задается расположение диаграммы. В верхней части вкладки в разделе **Placement** указывается, где будет расположена диаграмма.

Если выбрана опция **Once per report** (располагать только в одном месте отчета), диаграмма может быть расположена в заголовке (**Header**) или подвале (**Footer**) отчета. Если выбрана опция **For each**, диаграмма будет расположена в заголовке или подвале группы.

Четыре кнопки, расположенные в левой части вкладки, позволяют выбрать тип данных для диаграммы.

1. **Advanced** — данные, содержащиеся в секции **Details** — поля базы данных, формулы и поля **Running Total**.
2. **Group** — агрегатные данные, содержащиеся в полях **Summary**, например количество клиентов в каждом регионе.
3. **Cross-Tab** — агрегатные данные из таблиц **Cross-Tab**.
4. **OLAP** —данные из OLAP-источников.

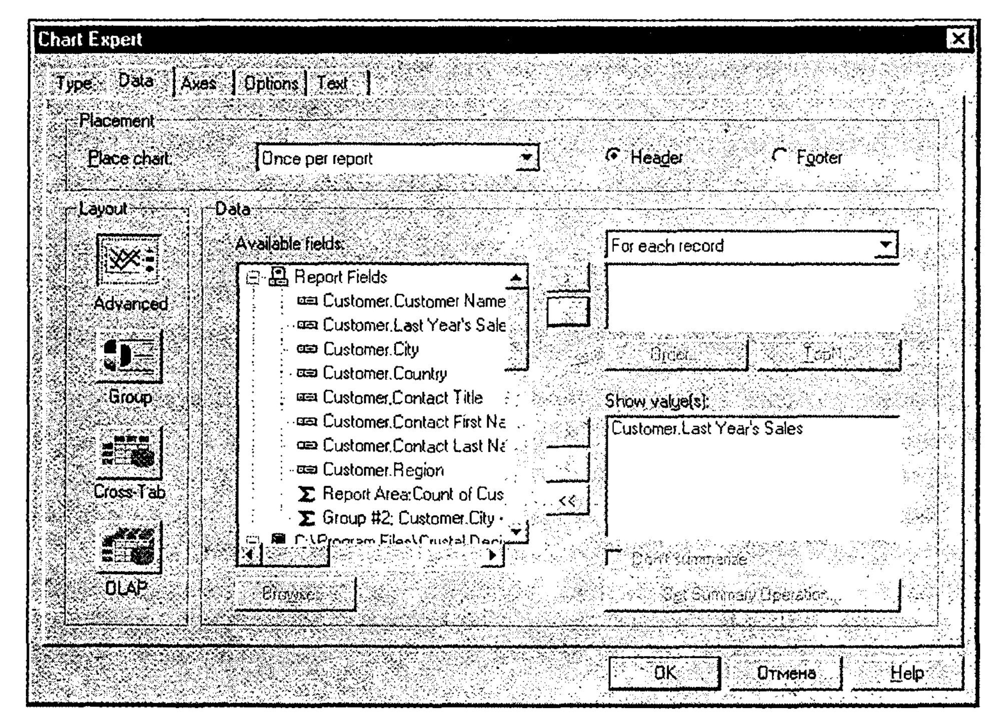


Рис. 4.2. Вкладка **Data** мастера отчетов **Chart Expert**

При создании диаграмм, основанных на агрегатных значениях, сохраняется возможность "высверливания" (**Drill Down**) данных. Предположим, создан отчет о продажах. Данные сгруппированы по странам, в отчет включено поле **Summary** с общим объемом продаж по стране. На основе агрегатных данных построена диаграмма типа **Pie**. Тогда для просмотра детальной информации можно перенести указатель на сегмент круга, при этом будет показан маркер в виде лупы, и дважды щелкнуть. Такая же возможность имеется и для подписей (**Legend**) к диаграмме. Более подробно отчеты **Drill Down** будут рассмотрены далее.

Содержимое правой части вкладки зависит от выбранного типа данных. Так, если выбран тип **Group**, то для построения диаграммы нужно указать поле отчета, в котором содержится аргумент — раскрывающийся список **On change of** — и поле, в котором содержится значение функции — раскрывающийся список **Show**.

Для отчетов на основе таблиц **Cross-Tab** рекомендуется выбирать тип диаграммы **3D Riser**. В этом случае в раскрывающемся списке **Sybdivided by** следует указать поле, в котором содержится второй аргумент.

На рис. 4.2 показан вид вкладки **Data** для типа данных **Advanced**. В этом случае диаграмма строится на основе полей, расположенных в секции

Details. В списке доступных полей необходимо выбрать поле аргумента или установить опцию **For each record**. В этом случае аргументом будет номер строки отчета. Затем в список **Show values** следует внести одно или несколько полей функций. При использовании типа данных **Advanced** рекомендуется выбирать тип диаграммы **Line** или **Area**.

Вкладки **Axes** и **Options** появляются, если на вкладке **Type** выключена опция **Automatically set char options**. На вкладке **Axes** (рис. 4.3) можно установить координатную сетку (**Show gridlines**) и масштабировать значения аргументов и функции (**Data values**). По умолчанию масштабирование значений аргументов и функции производится автоматически.

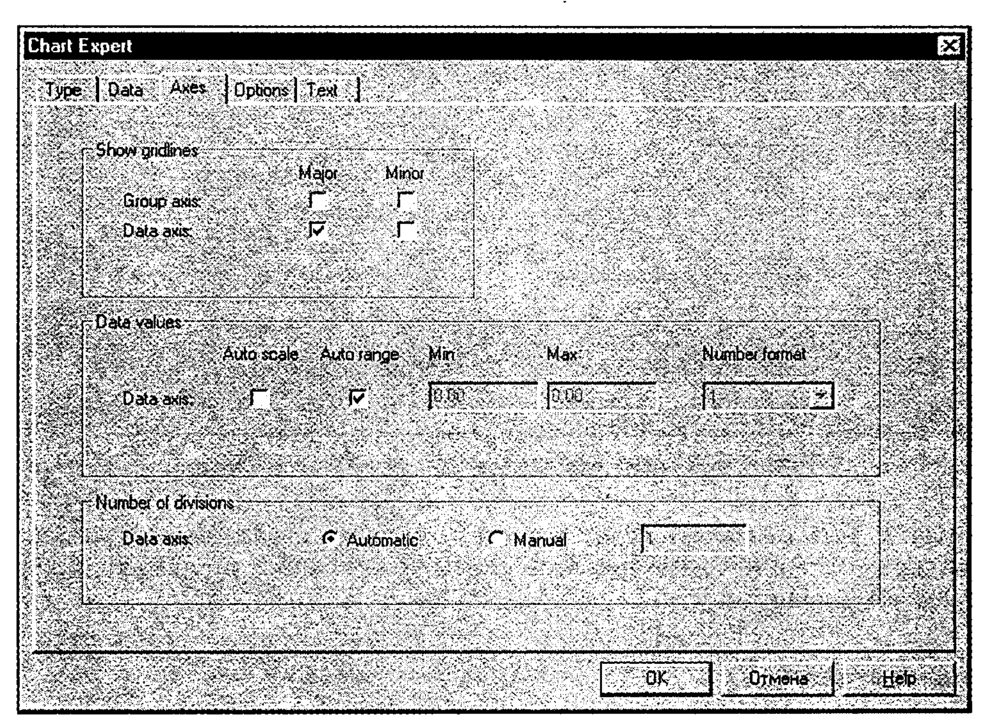


Рис. 4.3. Вкладка **Axes** мастера отчетов **Chart Expert**

Вкладка **Options** (рис. 4.4) позволяет задать дополнительные опции отображения диаграммы. Так, в разделе **Chart color** можно включить или отключить цвет диаграммы. Если в разделе **Data points** включить опцию **Show label** или **Show value**, то на диаграмме будет отображаться наименование поля функции или его значение. Опция **Show Legend** позволяет добавить подписи к элементам диаграммы. Например, если каждому городу на диаграмме соответствует фигура определенного цвета, в **Legend** будут включены подписи, которые свяжут цвет с городом. Если включить опцию **Transparent**

background, то это сделает диаграмму "прозрачной", другими словами, поля отчета будут служить для нее фоном.

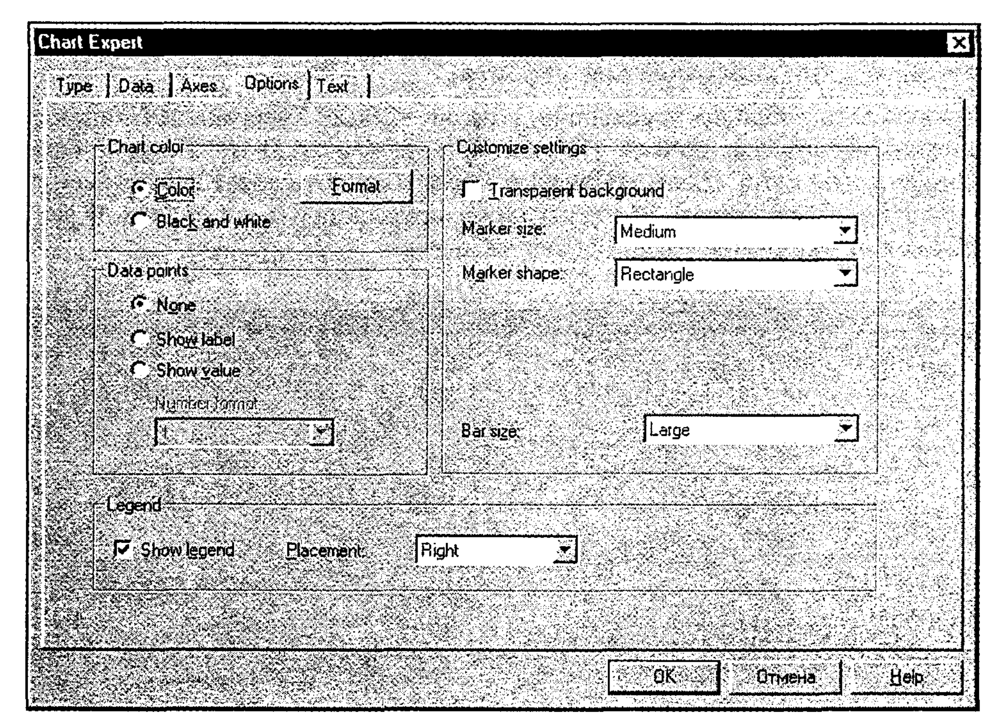


Рис. 4.4. Вкладка **Options** мастера отчетов **Chart Expert**

Вкладка **Text** позволяет создать текстовые поля, содержащие заголовки, подписи к осям и другие пометки. Здесь можно также установить шрифт для каждой подписи.

Упражнение 4.1

1. Откройте отчет, созданный в упражнении 3.5.
2. Удалите группировку по странам.
3. При помощи **Select Expert** задайте условие выборки `Customer.Country equal to "Canada"`.
4. Вставьте диаграмму типа **Pie**, отображающую количество клиентов в городах.
5. Результат выполнения упражнения представлен на рис. 4.5. Справа видны подписи (**Legend**).
6. Сохраните отчет.

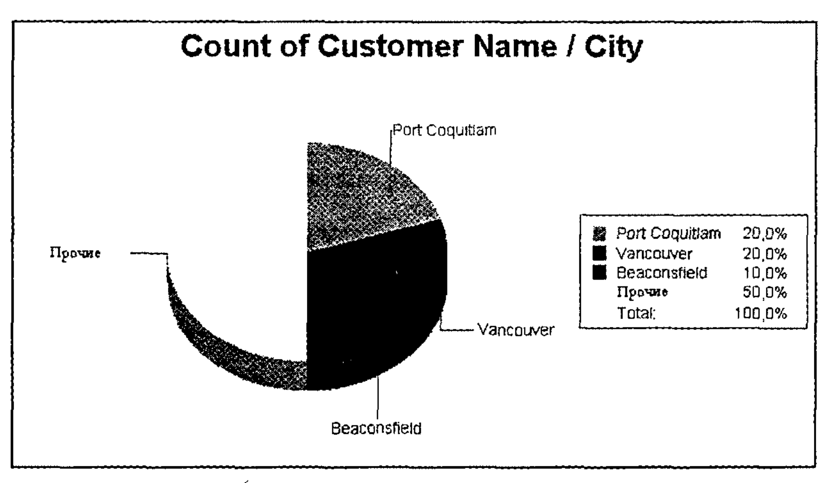


Рис. 4.5. Пример диаграммы типа **Pie**

4.2. Сложное форматирование диаграмм

После создания диаграммы при помощи **Char Expert**, ее можно отформатировать более тщательно. Для сложного форматирования диаграмм Crystal Reports 9 содержит специальный набор диалоговых окон, которые вызываются из контекстного меню. Для вызова контекстного меню следует переместить указатель на диаграмму и щелкнуть правой кнопкой мыши.

Пункт контекстного меню **Format Chart** вызывает диалоговое окно **Format Editor**. Это окно во многом похоже на подобное окно, служащее для форматирования полей (рис. 2.15). С его помощью можно задать основные свойства диаграммы на вкладке **Common**, создать рамку, используя вкладку **Border**, или создать гиперссылку на вкладке **Hyperlink**. Пункт контекстного меню **Size and Position** вызывает диалоговое окно **Object Size and Position** (рис. 4.6), в котором можно изменить размер и расположение диаграммы в секции отчета.

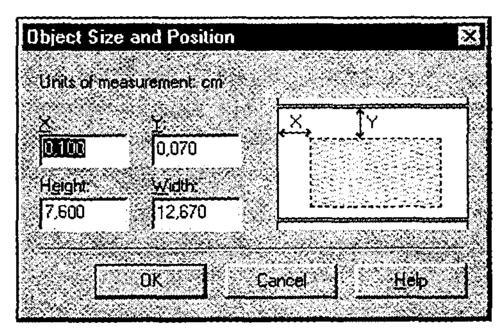


Рис. 4.6. Диалоговое окно **Object Size and Position**

С помощью команд меню **Chart Options | General**, **Chart Options | Titles** и **Chart Options | Template** вызываются дополнительные диалоговые окна форматирования.

Диалоговое окно **Chart Options** (рис. 4.7) можно также вызвать с помощью команды каскадного меню **Chart Options | General**.

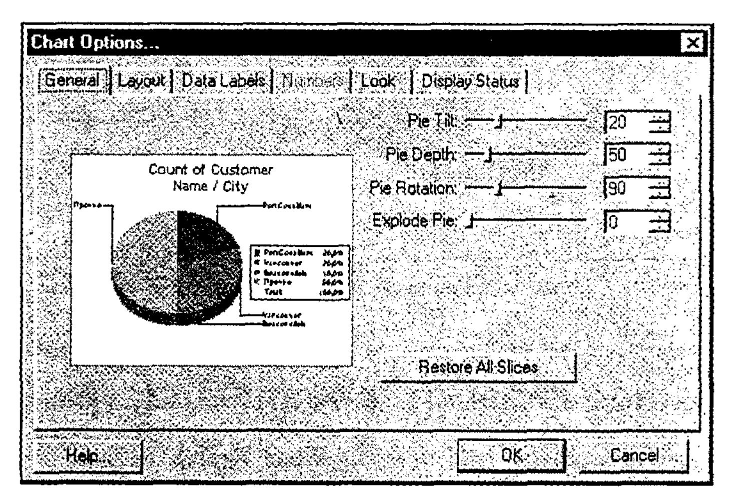


Рис. 4.7. Диалоговое окно **Chart Options**

Вкладка **General** является контекстной, т. е. ее содержимое зависит от типа диаграммы, и позволяет установить визуальные свойства диаграммы. Так, для типа **Pie** доступны свойства:

- **Tilt** задает угол зрения;
- **Dept** — высоту сегментов;
- **Rotation** — угол поворота;
- **Explode** — расстояние между сегментами;
- **Hole Size** — толщину кольца.

Вкладка **Layout** позволяет изменить подтип диаграммы, например, для типа **Pie** выбрать между кругом и кольцом. Вкладка **Data Labels** позволяет ввести подписи к сегментам или полоскам. На вкладке **Numbers** можно изменить формат изображения чисел. Вкладки **Look** и **Display Status** позволяют форматировать подписи к диаграмме, например, **Legend** или подписи к сегментам.

Команда меню **Chart Options | Titles** вызывает диалоговое окно **Titles** (рис. 4.8), в котором можно отредактировать заголовок, подзаголовок диаграммы и примечание.

При создании отчета высокого качества настройка диаграммы может быть довольно трудоемка. Для повторного использования настроек Crystal Reports 9

позволяет сохранить шаблон диаграммы (не путать с шаблоном отчета). Любую диаграмму можно сохранить как шаблон. Для этого следует щелкнуть правой кнопкой на диаграмме и выбрать в контекстном меню пункт **Save as Template**. Кроме того, Crystal Reports 9 содержит библиотеку готовых шаблонов. Чтобы воспользоваться шаблоном, нужно выполнить команду контекстного меню **Chart Options | Template**. Возникает диалоговое окно **Choose a Chart Type** (рис. 4.9). Вкладка **Gallery** содержит стандартные шаблоны, вкладка **Custom** — дополнительные, в том числе созданные пользователем.

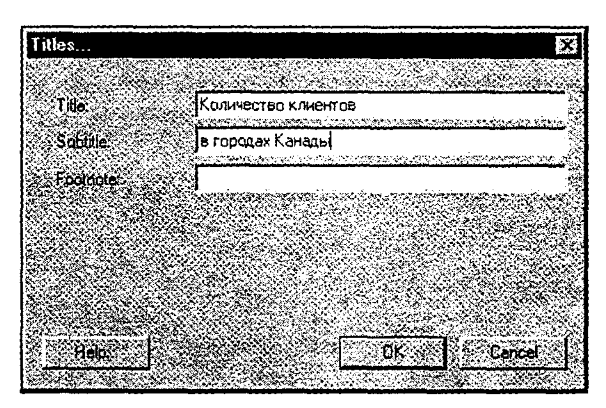


Рис. 4.8. Диалоговое окно **Titles**

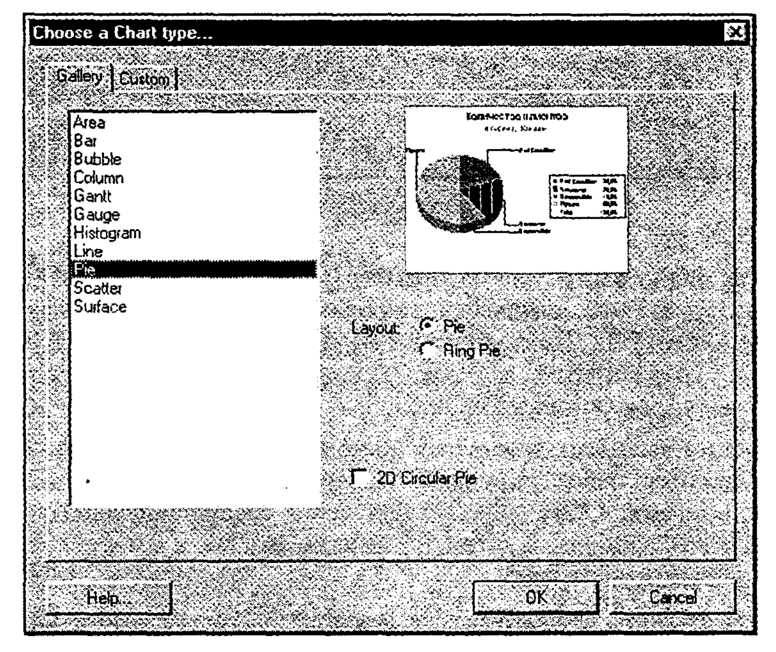


Рис. 4.9. Диалоговое окно **Choose a Chart Type**

Для форматирования отдельного элемента диаграммы, например сегмента круга или прямоугольника, используются диалоговые окна **Series Options** и **Formatting**. Для вызова этих окон следует переместить указатель на элемент диаграммы, щелкнуть правой кнопкой мыши и выполнить команду контекстного меню **Chart Options — Series** и **Chart Options | Selected Item** соответственно.

Диалоговое окно **Series Options** (рис. 4.10) позволяет изменить подпись и расположение фрагмента диаграммы, например, изменить расстояние до центра сегмента круга (опция **Detach Slice** на вкладке **General**). Кнопка **Delete Slice** служит для удаления выделенного фрагмента диаграммы, кнопка **Restore** — восстановления удаленного фрагмента. Вкладка **Data Labels** содержит радиокнопки, позволяющие скрыть или показать подписи фрагментов диаграммы, а также изменить их расположение.

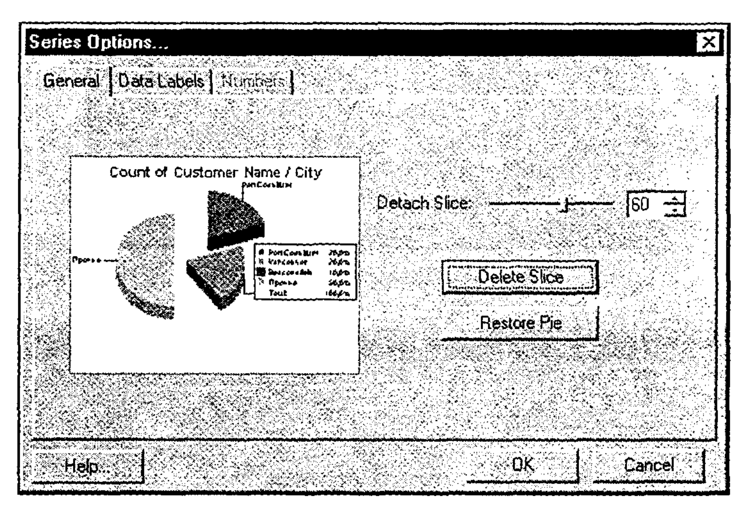


Рис. 4.10. Диалоговое окно **Series Options**

На вкладках диалогового окна **Formatting** (рис. 4.11) можно изменить шрифт объекта (вкладка **Font**), стиль и цвет обрамляющей линии (вкладка **Line**) и заливку (вкладка **Fill**). Помимо палитры цветов, Crystal Reports 9 содержит библиотеки градиентных заливок (кнопка **Gradient**), текстур (кнопка **Texture**) и рисунков (кнопка **Picture**).

Кнопка **Picture** вызывает диалоговое окно **Choose a Picture** (рис. 4.12), в котором содержится набор рисунков. Набор рисунков можно пополнить из библиотек Microsoft Office clip art. Для этого следует сначала щелкнуть на кнопке **Advanced Options**, а затем на кнопке **Browse** и выбрать файлы библиотеки с расширением wmf.

Выбранный рисунок отобразится на редактируемом элементе. Пример диаграммы с отображением рисунков на элементах приведен на рис. 4.13.

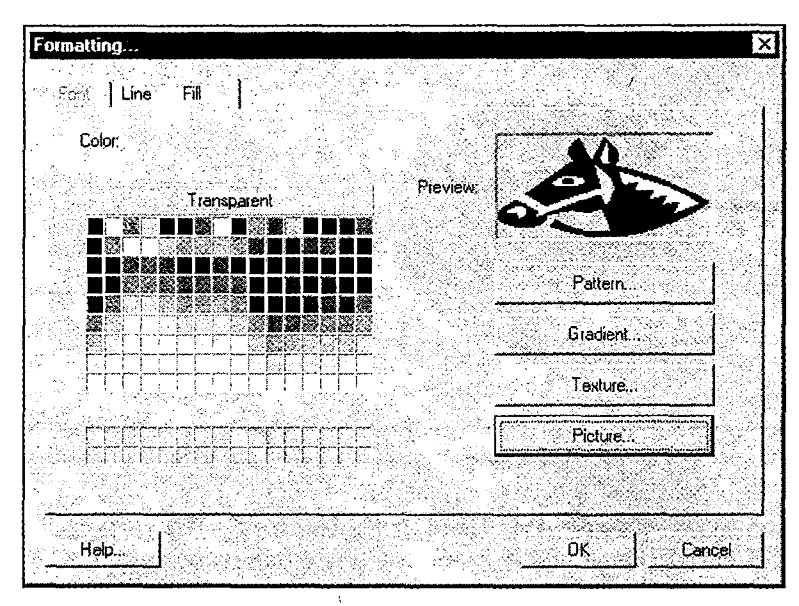


Рис. 4.11. Диалоговое окно **Formatting**

Choose a Picture...
<< Presets Only
AN01124_
No Flipping
Flip Vertical
Flip Horizontal
Flip Both
Picture File Name
Browse...
Help...
OK
Cancel

Рис. 4.12. Диалоговое окно **Choose a Picture**

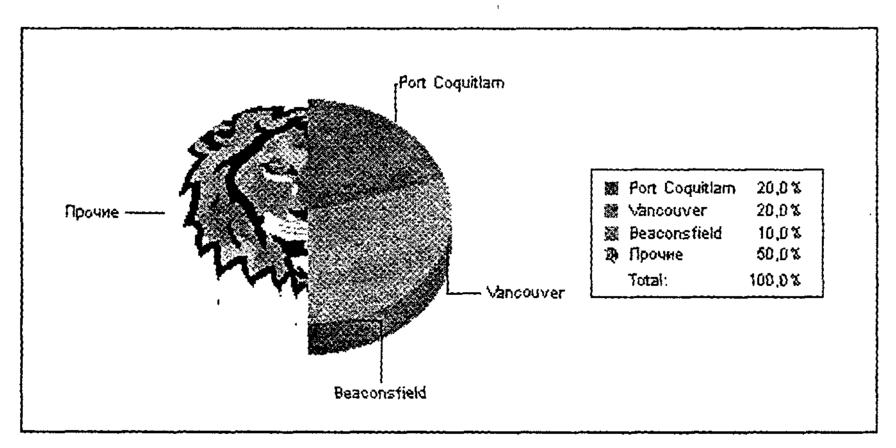


Рис. 4.13. Пример диаграммы с отображением рисунков на элементах

Диалоговое окно **Numeric Axis Grids & Scales** (рис. 4.14) вызывается командой меню **Chart Options | Grid** и служит для форматирования координатной сетки. Заметим, что оно недоступно для диаграмм типа **Pie**. Для трехмерных диаграмм диалог имеет два ряда вкладок — по горизонтали и по вертикали.

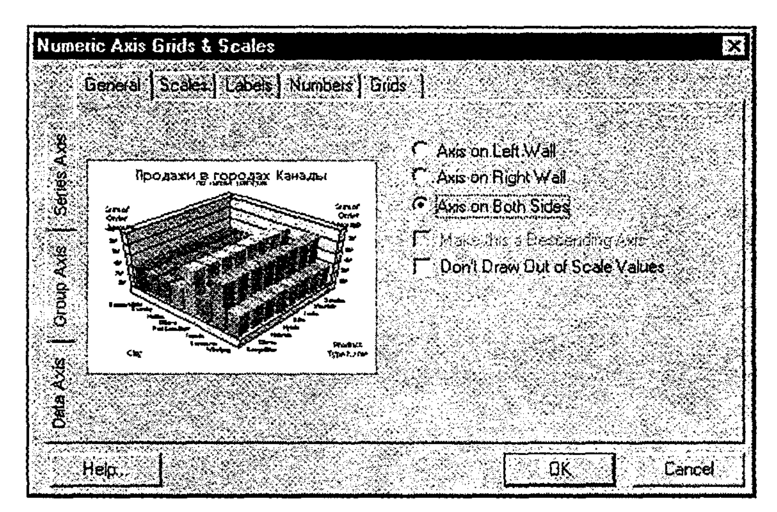


Рис. 4.14. Диалоговое окно **Numeric Axis Grids & Scales**

Вкладки, расположенные по вертикали, позволяют выбрать тип координатной сетки: **Data Axis** — координатная сетка по вертикали, **Group Axis** и **Series Axis** — по двум другим координатам. Горизонтальная вкладка **Grids** служит для включения и выключения координатной сетки. Вкладки **Labels** и **Numbers** позволяют ввести подписи к осям и задать их формат. Во вкладке **Scales** задается масштаб данных на осях.

Рис. 4.15. Диалоговое окно **Choose a Viewing Angle**

Диалоговое окно **Choose a Viewing Angle** позволяет выбрать точку зрения на трехмерную диаграмму. Рис. 4.15 показывает вид диалогового окна **Choose a Viewing Angle** после щелчка на кнопке **Advanced Options**. Кнопка исчезает, и появляется инструмент изменения угла зрения. Вкладка **Rotate** позволяет вращать изображение, вкладка **Pan** — изменять "удаление" до изображения, **Walls** — менять толщину стенок координатных плоскостей, **Move** — перемещать изображение.



Пакет Crystal Reports 9 содержит типовые отчеты с диаграммами — Chart.rpt и Employee Sales.rpt. Их можно найти в каталогах Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Reports\Feature Examples и Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Reports\General Business соответственно.

4.3. Вставка географических карт

Crystal Reports 9 позволяет включить в отчет географические карты и связать с ними данные отчета. Карты могут быть отформатированы, кроме того, при работе с картами сохраняется возможность "высверливания" (drill-down) данных. Привязка данных к географическим картам делает отчет нагляднее,

эффективно выявляет тенденции и, тем самым, дает возможность произвести более качественный анализ.

Для внесения в отчет географической карты следует щелкнуть на кнопке вставки географической карты в панели инструментов для вставки объектов в отчет (табл. 1.3) или выполнить команду меню **Insert | Map**. Появляется диалоговое окно **Map Expert**, которое содержит три вкладки: **Data**, **Type** и **Text**.

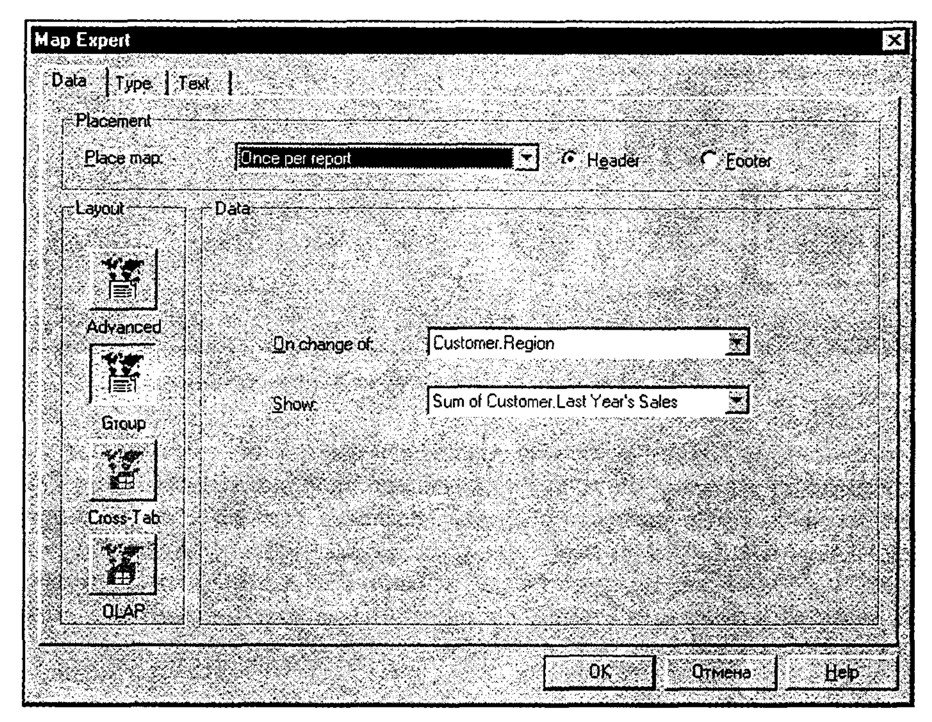


Рис. 4.16. Вкладка **Data** диалогового окна **Map Expert**

На вкладке **Data** (рис. 4.16) задается тип данных, которые будут связаны с географической картой. Если выбрана опция **Once per report** (располагать только в одном месте отчета), карта может быть расположена в заголовке (**Header**) или подвале (**Footer**) отчета. Если выбрана опция **For each**, то карта будет расположена в заголовке или подвале группы. Так же как и диаграмма, географическая карта может быть связана:

- кнопкой **Group** с агрегатными данными, содержащимися в полях **summary**;
- кнопкой **Advanced** с данными, содержащимися в секции **Details**;
- кнопкой **Cross-Tab** с агрегатными данными из таблиц **Cross-Tab**;
- кнопкой **OLAP** — с данными из OLAP-источников.

Если географическая карта связана с данными, содержащимися в секции **Details**, на вкладке появляется несколько полей и списков. С помощью кнопки > поля данных из списка доступных полей (**Available Fields**) заносятся в списки полей **On change of**, **Geographic field** и **Map value**, а с помощью кнопки < удаляются из этих списков. Опция **On change of** используется как условие для суммирующего поля, значение поля **Map value** будет использовано для отображения на карте.

В правой части вкладки нужно указать поле, содержащее географические названия (**Geographic field**), например поле, содержащее названия стран, областей, городов и т. д.

Crystal Reports 9 поддерживает стандарт геоинформационной системы **MapInfo**. При инсталляции Crystal Reports 9 устанавливается утилита **Geodictionary Manager** (рис. 4.17) и автоматически регистрируется набор географических карт. Значение поля **Geographic field** должно совпадать со значением поля таблицы географических карт. Для регистрации дополнительной географической карты необходимо щелкнуть на кнопке **Пуск** (**Start**), затем выбрать пункт меню **Выполнить** (**Run**) и в диалоговом окне **Запуск программы** (**Run**) указать файл \Program Files\MapInfo MapX\ Program\Migm30.exe.

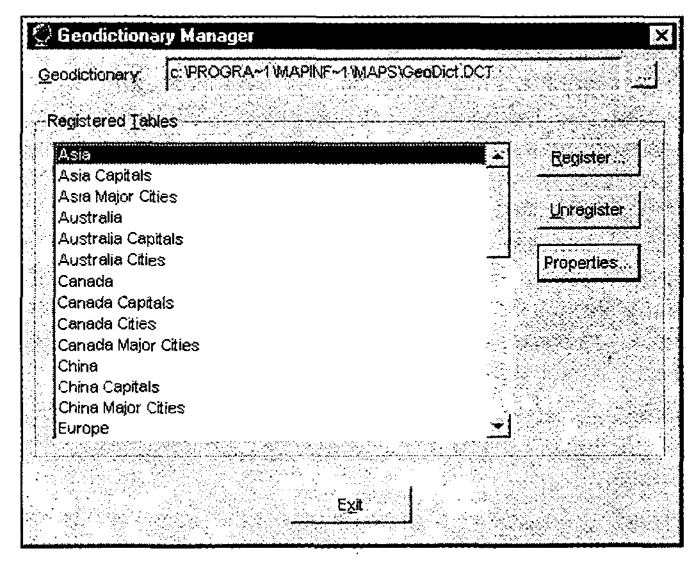


Рис. 4.17. Диалоговое окно утилиты **Geodictionary Manager**

С помощью утилиты **Geodictionary Manager** необходимо указать файл Geodictionary с расширением dct. Если щелкнуть на кнопке **Properties**, откроется диалоговое окно **Table Properties** (рис. 4.18), в котором можно просмотреть поля, содержащие географические названия, и изменить свойства таблицы.

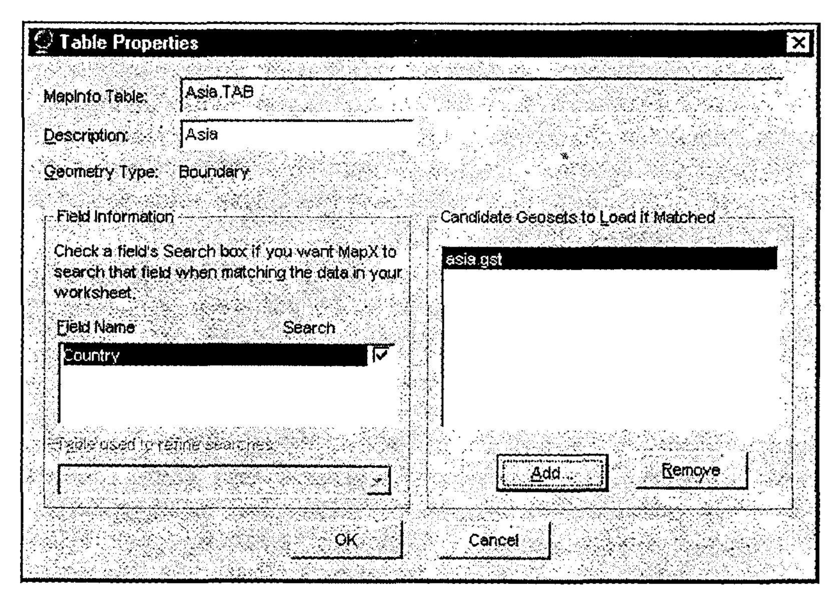


Рис. 4.18. Диалоговое окно **Table Properties**

Вкладка **Type** диалога **Map Expert** (рис. 4.19) позволяет выбрать один из пяти возможных типов карты.

- **Ranged** — географические области закрашиваются с интенсивностью, пропорциональной отображаемой величине. Например, если создается карта, показывающая количество людей, проживающих в каждом регионе, то можно ввести условие, что каждому оттенку будут соответствовать интервалы 0—5000, 5001—10000, 10001—15000, 15001—20000 и более 20000 человек. Для закраски областей удобно выбрать оттенки одного цвета, например, регионы с наибольшим населением показать темно-зеленым цветом, с наименьшим — светло-зеленым. Количество оттенков, как и количество учитываемых групп, можно изменять. Есть четыре алгоритма распределения интервалов:
  - **Equal count** — интервалы распределяются так, чтобы число регионов в одном интервале было примерно одинаковым, сами интервалы будут различными;
  - **Equal ranges** — каждому интервалу соответствует одинаковый количественный диапазон, например 0—5000, 5001—10000, 10001—15000 и т. д.;
  - **Natural break** — алгоритм, по которому минимизируется разница между суммарным значением и средним значением от суммарного для каждого интервала;
  - **Standard deviation** — интервалы рассчитываются на основе среднеквадратичного отклонения.

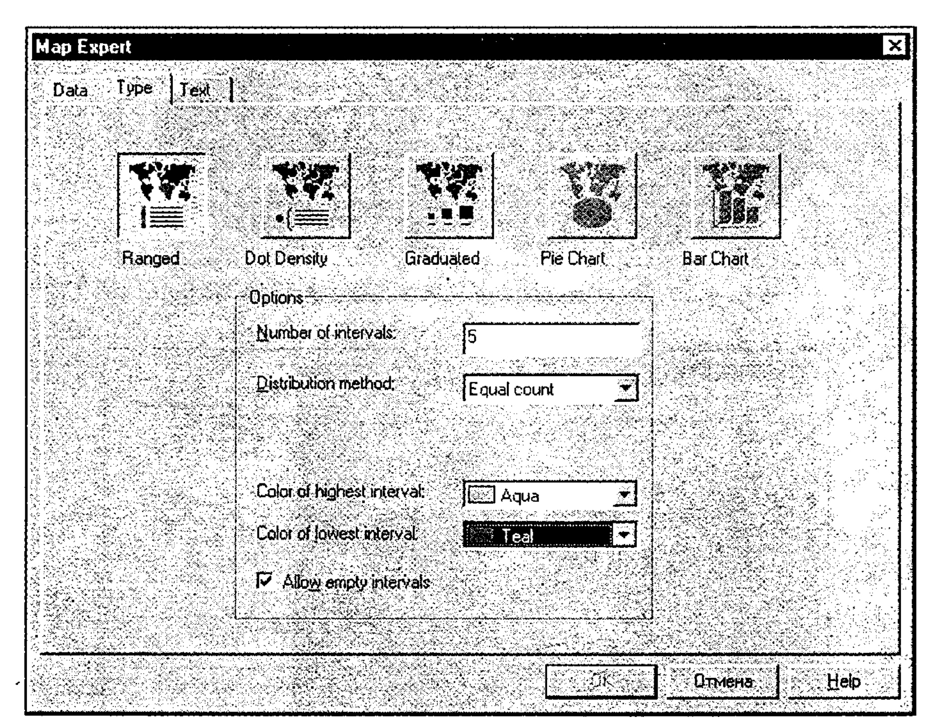


Рис. 4.19. Вкладка **Type** диалога **Map Expert**

- **Dot Density** — области заливаются узором из точек. Количество точек пропорционально отображаемой величине. Чем больше величина, тем большая плотность точек будет отображаться в области на географической карте.
- **Graduated** — в каждой области показывается один символ. Размер символа пропорционален отображаемой величине. Тип символа можно изменить (кнопка Custom), это может быть любая литера из любого шрифта.
- **Pie Chart** — на фоне карты показывается диаграмма типа **Pie**.
- **Bar Chart** — на фоне карты показывается диаграмма типа **Bar**.

Вкладка **Text** служит для внесения заголовка географической карты и изменения подписей (**Legend**) к карте.

Если щелкнуть на кнопке **OK** в редакторе **Map Expert**, географическая карта будет включена в отчет (рис. 4.20 и 4.21).

Для редактирования географической карты служит контекстное меню (табл. 4.1), которое можно вызвать, щелкнув на карте правой кнопкой мыши.

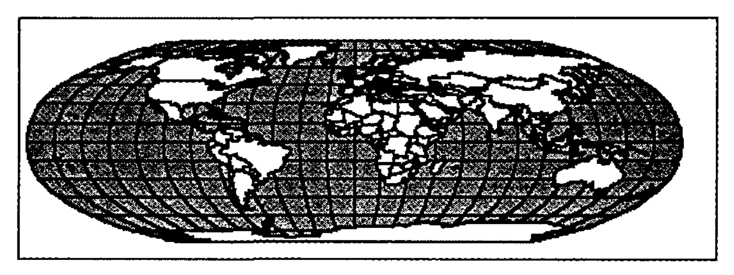


Рис. 4.20. Вид географической карты в режиме **Design**

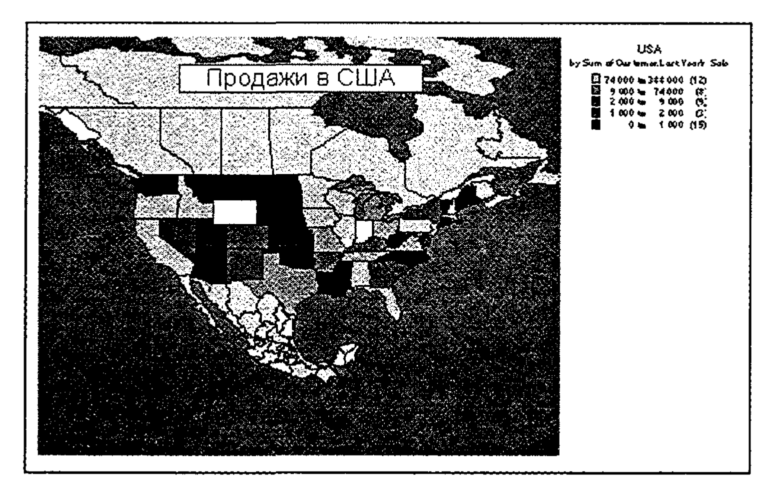


Рис. 4.21. Вид географической карты в режиме **Preview**

Таблица 4.1. *Пункты контекстного меню редактирования географической карты*

Пункт меню	Описание команды меню
Map Expert	Вызов диалогового окна **Map Expert** (рис. 4.16)
Format Map	Вызов диалогового окна **Format Editor**. Диалоговое окно **Format Editor** аналогично окну для форматирования полей, которое было рассмотрено в *разд. 2.2* (рис. 2.15). С его помощью можно отформатировать карту, изменить ее рамки и установить гиперссылку
Size and Position	Вызов диалогового окна **Object Size and Position** (рис. 4.6), служащего для изменения размера и расположения карты
Select Mode	Переход в режим выбора карты
Zoom In	Увеличение масштаба. Для увеличения масштаба следует выбрать пункт меню **Zoom In** и затем щелкнуть на карте
Zoom Out	Уменьшение масштаба. Для уменьшения масштаба следует выбрать пункт меню **Zoom Out** и щелкнуть на карте

***Таблица 4.1** (окончание)*

Пункт меню	Описание команды меню
Pan	Перемещение карты. Для перемещения карты следует выполнить команду меню **Pan** и переместить карту методом Drag&Drop
Center Map	Центровка карты. Для центровки карты следует выбрать пункт меню **Center Map** и щелкнуть на карте
Title	Вызов диалогового окна **Change Map Title** для изменения заголовка карты
Type	Вызов диалогового окна **Customize Map** для изменения опций закрашивания зон карты
Layers	Вызов диалогового окна **Layer Control** для изменения слоев карты
Resolve Mismatch	Вызов диалогового окна **Resolve Map Mismatch** для разрешения противоречий между данными географического поля отчета и данными карты
Map Navigator	Вызов окна **Map Navigator**

Для изменения заголовка карты в контекстном меню выбрать пункт **Title**. В появившемся диалоге **Change Map Title** (рис. 4.22) следует изменить заголовок карты и щелкнуть на кнопке **OK**.

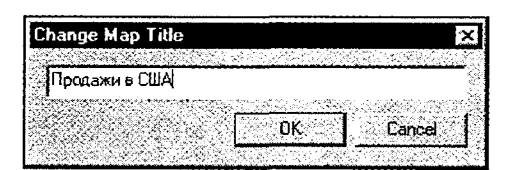


Рис. 4.22. Диалоговое окно **Change Map Title**

Чтобы изменить тип и настройки отображения карты, можно вернуться в диалоговое окно **Map Expert**, однако существует более короткий путь. Если в контекстном меню выбрать пункт **Type**, появится диалоговое окно **Customize Map** (рис. 4.23), в верхней части которого расположен раскрывающийся список, содержащий типы карт. Содержание нижней части окна зависит от выбранного типа. Так, для типа **Ranged** можно изменить опции закрашивания зон карты, для типа **Dot Density** — размер точек, составляющих узор, для типа **Graduated** — стиль символов.

Географическая карта в формате **MapInfo** содержит несколько слоев. Слой карты может, например, содержать города, или реки, или границы областей. Диалоговое окно **Layer Control** (рис. 4.24) позволяет изменить состав и очередность слоев карты.

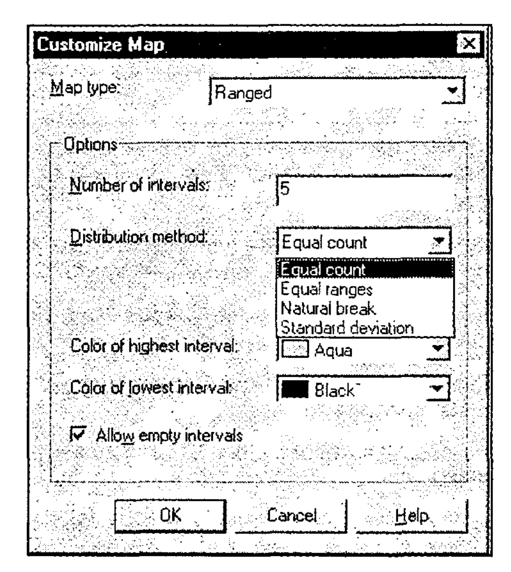


Рис. 4.23. Диалоговое окно **Customize Map**

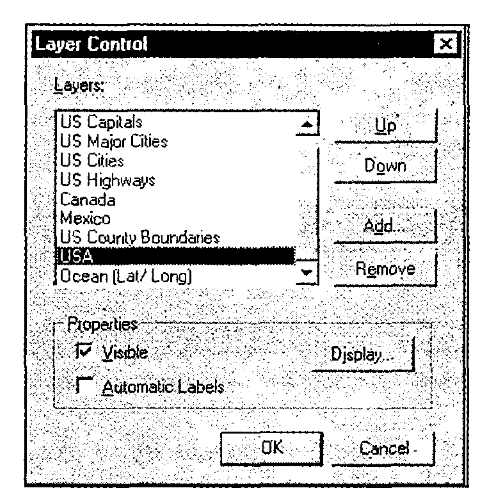


Рис. 4.24. Диалоговое окно **Layer Control**

Содержимое географического поля может не совпадать с данными географической карты. Например, в таблице карты может содержаться название страны United Kingdom, а в отчете может соответствовать ему значение UK. Для разрешения противоречий используется диалоговое окно **Resolve Map Mismatch**, содержащее две вкладки — **Change Map** и **Resolve Mismatch**. Зарегистрированная с помощью утилиты **Geodictionary Manager** карта, которую подключает по умолчанию Crystal Reports 9, может оказаться неподходящей для создаваемого отчета. Вкладка **Change Map** (рис. 4.25) позволяет переоп-

ределить подключенную карту. Для переопределения карты следует переместить указатель на строку с наименованием карты в списке **Available Map**. В окне **Selected Map** отображается выбранная карта.

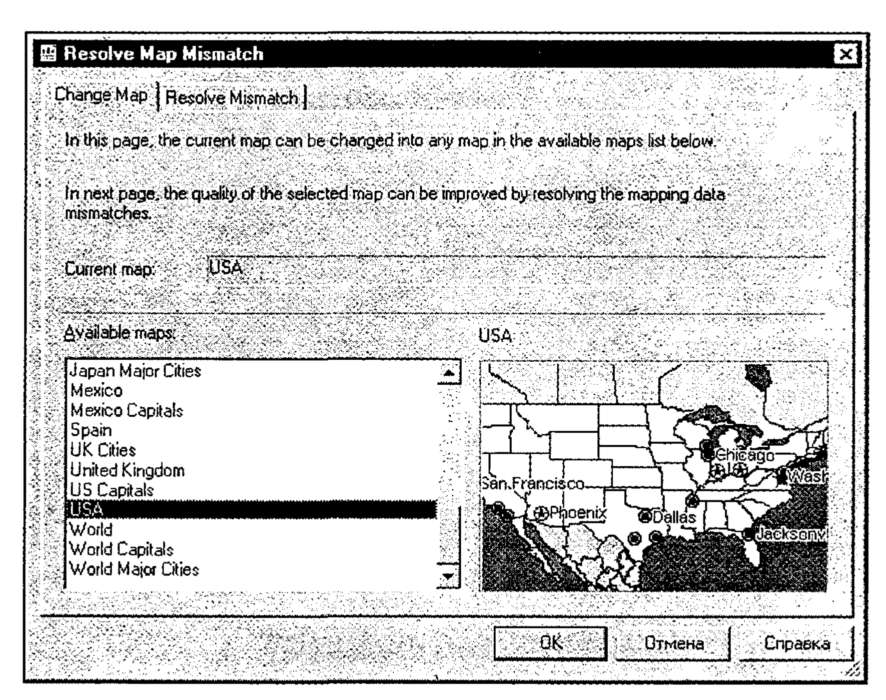


Рис. 4.25. Вкладка **Change Map** диалогового окна **Resolve Map Mismatch**

Во вкладке **Resolve Mismatch** (рис. 4.26) показываются географические имена, не соответствующие значению полей отчета. Для разрешения противоречия в списке **Assign this Field Name** следует выбрать значение поля отчета, затем выбрать значение из списка **To this Map Name**, после чего щелкнуть на кнопке **Match**. Соответствие полей показывается в списке **Matched results**. После разрешения всех противоречий следует щелкнуть на кнопке **OK**.

Увеличение масштаба карты (**Zoom In**) и перемещение (**Pan**) приводит к тому, что в отчете показывается только часть карты, причем иногда оказывается сложно определить, какая именно часть. Для навигации по карте используется окно **Map Navigator** (рис. 4.27). Этот инструмент позволяет взглянуть в мелком масштабе на положение отображаемого фрагмента карты относительно целой карты. Текущее положение обозначается прямоугольником на целой карте. **Map Navigator** поддерживает перемещение карты. Для перемещения достаточно сдвинуть прямоугольник в окне **Map Navigator** методом Drag&Drop. Скрыть **Map Navigator** можно, выключив опцию **Map Navigator** в контекстном меню.

Resolve Map Mismatch

Change Map | Resolve Mismatch

Assign this Field Name:

To this Map Name:

Afghanistan
Albania
Algeria
American Samoa
Andorra
Angola

Match

Unmatch

Matched results:

Field Name	Map Name
Australia	Australia
Austria	Austria
Bangladesh	Bangladesh
Brazil	Brazil
Canada	Canada
China	China
Costa Rica	Costa Rica

Selected map:

OK | Cancel | Help

Рис. 4.26. Вкладка **Resolve Mismatch** диалогового окна **Resolve Map Mismatch**

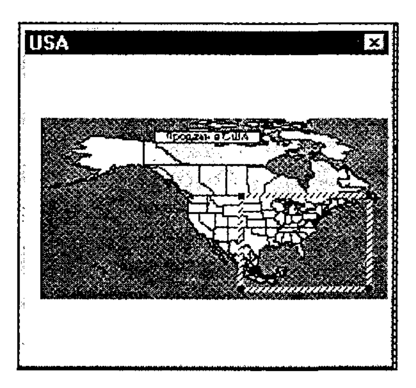
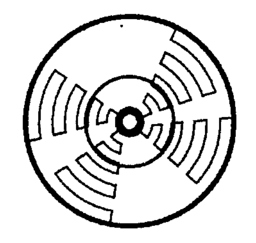


Рис. 4.27. Окно **Map Navigator**

Глава 5



Использование формул

5.1. *Formula Workshop* — среда создания формул

В отчетах часто возникает необходимость отображать поля, которые не содержатся в базе, но могут быть вычислены. Для этого Crystal Reports 9 содержит эффективный инструмент — язык описания формул.

Чтобы создать формулу, достаточно в списке полей окна **Field Explorer** переместить указатель на строку **Formula Fields** и щелкнуть кнопкой мыши на кнопке создания объекта (табл. 2.1).

Появляется диалоговое окно **Formula Name** (рис. 5.1) для ввода имени формулы.

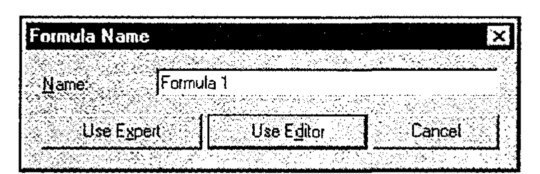


Рис. 5.1. Диалоговое окно **Formula Name**

В поле **Name** следует ввести имя формулы и щелкнуть на кнопке **Use Expert** или **Use Editor**. Появляется диалоговое окно **Formula Workshop** (рис. 5.2).

Formula Workshop представляет собой универсальную среду, позволяющую создавать или изменять следующие объекты:

- специальные функции отчета;
- специальные функции, размещенные в хранилище (Repository);
- формулы для вычислений и преобразований данных отчета;
- SQL-выражения;
- формулы для выборки данных;

- формулы для форматирования данных;
- формулы для создания специальных сообщений (Report Alert) отчета.

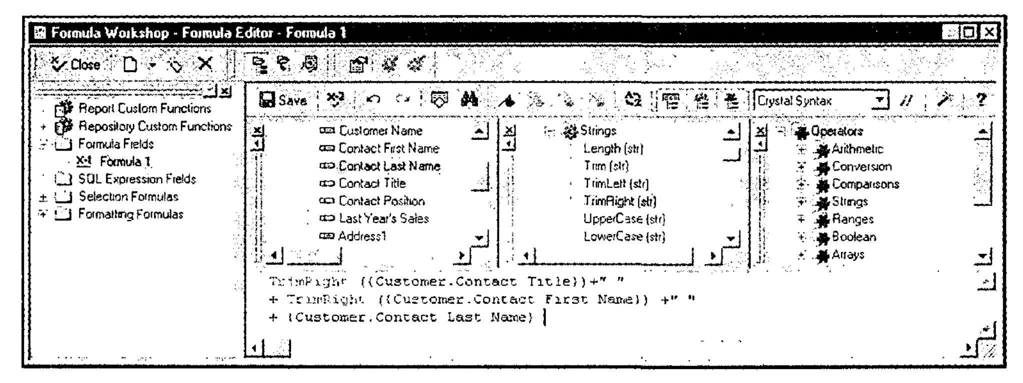


Рис. 5.2. Диалоговое окно **Formula Workshop** с окном **Formula Editor**

По умолчанию в левой части диалогового окна **Formula Workshop** расположен древовидный список, содержащий набор папок для каждого типа объектов, в верхней части — панель инструментов (табл. 5.1). Древовидный список функций и формул является перемещаемым.

Таблица 5.1.** Элементы управления панели инструментов диалогового окна **Formula Workshop

Элемент управления	Описание элемента
Close	Сохранение формулы и закрытие диалогового окна **Formula Workshop**. Перед закрытием окна проверяется синтаксис формулы
	Создание новой формулы, функции, определяемой пользователем или SQL-выражения. Раскрывающийся список позволяет выбрать тип создаваемого объекта
	Переименование формулы, функции, определяемой пользователем или SQL-выражения
	Удаление формулы, функции, определяемой пользователем или SQL-выражения
	Показать или скрыть список функций и формул
	Раскрыть или свернуть список функций и формул
	Показать или скрыть объекты в разделе формул форматирования (**Formatting Formulas**) списка функций и формул, которые не связаны с формулами форматирования

Таблица 5.1 (окончание)

Элемент управления	Описание элемента
	Переключение между окнами **Custom Function Editor** и **Custom Function Properties**
	Открыть диалоговое окно **Opens the Add Custom Function to Repository** для внесения специальной функции в хранилище
	Внести выбранную специальную функцию из хранилища в отчет

Если создается или изменяется формула или SQL-выражение, то в правой нижней части содержится окно **Formula Editor**, если же создается или изменяется специальная функция, то окно **Formula Expert**.

Окно **Formula Editor** (рис. 5.2) содержит панель инструментов и четыре окна. Элементы управления панели инструментов **Formula Editor** приведены в табл. 5.2. Окно полей содержит древовидный список полей отчета, специальных полей, агрегатных полей, формул и полей базы данных. Окно функций содержит древовидный список функций. Функции представляют собой предопределенные процедуры, возвращающие значение. В списке функции отсортированы по типам. Окно операторов содержит список операторов, использующихся в формулах.

В нижней части **Formula Editor** размещается окно с текстом формулы. Если дважды щелкнуть по наименованию формулы поля, наименованию функции или наименованию оператора в соответствующем списке, то текст формулы выводится в это окно для редактирования.

Если установить указатель на какой-либо фрагмент формулы в нижней части окна и нажать комбинацию клавиш <Ctrl>+<Пробел>, то появляется список доступных функций и служебных слов, из которых можно выбрать необходимое.

Все окна и панель инструментов **Formula Editor** перемещаемые — можно расположить их так, как удобно.

*Таблица 5.2. Элементы управления панели инструментов окна **Formula Editor***

Элемент управления	Описание элемента
Save	Сохранение формулы. Перед закрытием окна проверяется синтаксис формулы
	Проверка синтаксиса. В случае обнаружения ошибки выдается сообщение

Таблица 5.2 (продолжение)

Элемент управления	Описание элемента
	Отмена действия (Undo)
	Повтор действия (Redo)
	Просмотр значения полей. Появляется редактор просмотра значения поля. Чтобы внести значение поля в текст формулы, достаточно дважды щелкнуть на соответствующей строке в списке значений
	Поиск. Поиск осуществляется в тексте формулы
	Установка меток. В тексте формулы меткой может быть помечена строка целиком. Метки в Formula Editor используются только для навигации
	Переход к последующей метке
	Переход к предыдущей метке
	Удаление всех меток
	Изменение порядка функций или операторов. Функции и операторы могут быть отсортированы как по типу, так и по алфавиту
	Включение и выключение окна полей
	Включение и выключение окна функций
	Включение и выключение окна операторов
Crystal Syntax	Выбор синтаксиса. Crystal Reports 9 поддерживает два синтаксиса для формул — традиционный синтаксис Crystal Reports и синтаксис Visual Basic. Один отчет может содержать формулы, написанные в разном синтаксисе, однако одна формула не может содержать фрагменты, написанные в разных стилях. В настоящей книге рассматривается только синтаксис Crystal Reports 9
//	Внесение комментария. Щелчок на этой кнопке приводит к тому, что в начало текущей строки добавляется два символа /, после чего данная строка считается комментарием *(разд. 5.2)*

***Таблица 5.2** (окончание)*

Элемент управления	Описание элемента
	Вызов окна **Formula Expert**
	Вызов справки

Окно **Formula Expert** (рис. 5.3) позволяет создавать, изменять и вносить специальные функции в отчет. В левой верхней части окна расположен древовидный список специальных функций. В поле **Summary** показывается описание функции, в поле **Return Type** — тип возвращаемого значения функции. Для внесения специальной функции в отчет необходимы следующие действия:

1. Переместить указатель в списке специальных функций на строку с требуемой функцией.
2. В списке **Functions Arguments** указать значения аргументов.
3. Щелкнуть на кнопке **Save**.

Щелчок на кнопке **Use Editor** открывает окно **Formula Editor**, причем в окно с текстом формулы добавляется специальная функция.

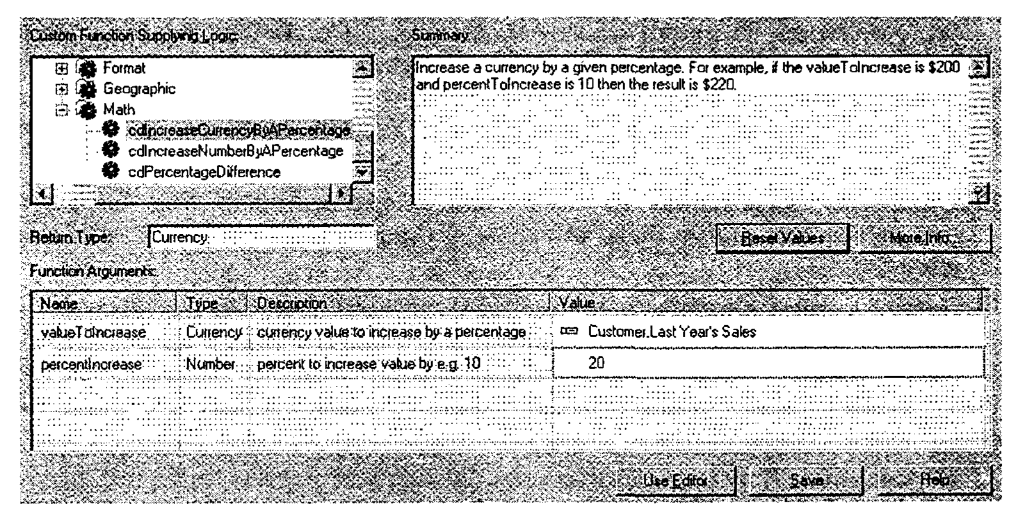


Рис. 5.3. Окно **Formula Expert**

5.2. Синтаксис формул

Как было указано ранее, Crystal Reports 9 поддерживает два типа вида синтаксиса — синтаксис Crystal и синтаксис Basic. Любая формула, написанная по правилам Cristal, может быть переписана по правилам Basic и наоборот. Отчет может содержать формулы обоих типов. Синтаксис Crystal традиционен для Crystal Reports, его поддерживают все предыдущие версии этого продукта. Синтаксис Basic введен только в версии Crystal Reports 8.0. Он похож на синтаксис Microsoft Visual Basic за исключением выражений, которые отвечают за управление отчетом. Ниже будет рассматриваться синтаксис Crystal.

Формулы Crystal Reports 9 могут содержать следующие компоненты.

- Поля отчета или поля базы данных, которые выбираются из списка полей или вносятся вручную. Эти компоненты заключаются в фигурные скобки: {Customer.Last Year's Sales}, {Customer.City}.
- Числа: 1, 2, 3.1416, здесь точка отделяет дробную часть действительного числа.
- Текст, заключенный в двойные кавычки: "Итого сумма по отчету".
- Операторы выбираются из окна операторов или вводятся вручную. Это могут быть арифметические операторы (/, *, +, -), булевы операторы (and, or), операторы сравнения (<, >,=) и другие.
- Функции, в том числе агрегатные и специальные: Sum (x, {customer.City}), cdIncreaseCurrencyByAPercentage ({Customer.Last Year's Sales}, 20), Trim (x). Аргументы функции заключаются в круглые скобки и разделяются запятой. Вторым аргументом агрегатной функции Sum(x, {Customer.City}) является поле, по которому произведено группирование. Функция может быть выбрана из списка функций или внесена вручную. Подробнее функции рассмотрены в *разд. 5.3.*
- Структуры управления: If, Select или оператор цикла For. Структуры управления могут быть выбраны из окна операторов или внесены вручную.
- Другие формулы отчета, например {@GrossProfit}, {@QUOTA}. Имя формулы заключается в фигурные скобки и начинается символом @. Формулы выбираются из окна полей или вносятся вручную. Несмотря на то, что в окне полей показывается имя текущей формулы, в тексте формулы его использовать нельзя.
- Переменные. Перед использованием переменные должны быть предварительно описаны в формуле. Crystal Reports 9 поддерживает переменные типа массив (array). Индекс массива заключается в квадратные скобки: MyArray[5].

Комментарий к формуле отделяется символами //.

В формуле могут также использоваться параметры, поля running total, поля SQL expression, специальные поля, суммирующие поля и имена групп. Эти элементы могут быть выбраны из списка полей или внесены вручную.

- Параметры заключаются в фигурные скобки. Имя параметра начинается со знака ?, например:

  ```
  {?my parameter field}.
  ```

- Имя поля running totals начинается со знака #:

  ```
  {#my running total}.
  ```

- Поля SQL expression начинаются со знака процента:

  ```
  {%my SQL expression}.
  ```

- Суммирующие поля и имена групп, похожие на вызовы функций:

  ```
  Sum({Orders.Order Amount}, {Orders.Ship Via})
  GroupName({Orders.Ship Via}).
  ```

Каждая формула должна возвращать значение одного из семи типов: Number, Currency, String, Boolean, Date, Time или DateTime. Crystal Reports 9 поддерживает дополнительные типы переменных, которые не могут быть возвращаемым значением формулы, это диапазон (range) и массив (array).

Синтаксис Crystal нечувствителен к регистру. Так, служебные слова Then, then и THEN эквивалентны. Это правило не относится к тексту, заключенному в двойные кавычки. Использование формул имеет следующие ограничения:

- максимальное число выполнения цикла не может превышать 30000;
- размер строки (константы, строковой переменной, возвращаемого значения функции) не может превышать 64K;
- размерность массива не может превышать 1000;
- текст формулы не может превышать 64K;
- максимальное число аргументов функций для функций с переменным числом аргументов, например Choose, не должно превышать 1000;
- год даты должен находиться в диапазоне 1—9999.

Ниже приведены простейшие формулы, содержащие поля базы данных.

Первая формула вычисляет количество дней, прошедших от заказа до отгрузки товара. Поле типа Date — Orders.Order Date содержит дату заказа, поле Orders.Ship Date — дату отгрузки товара. Формула возвращает целое число.

```
//A formula that uses database fields
{Orders.Ship Date} - {Orders.Order Date}
```

Вторая формула возвращает стоимость заказа, которая равна произведению цены за единицу товара на количество.

```
{Orders Detail.Unit Price} * {Orders Detail.Quantity}
```

Выражением (Expression) называется любая комбинация операторов, ключевых слов, функций и констант, возвращающих значение определенного типа, например,

```
2+2;
```

возвращает число 4, а

```
"Иван "+"Иванов";
```

возвращает строку "Иван Иванов".

Формула может содержать несколько выражений. В этом случае каждое выражение должно заканчиваться символом ;. Формула возвращает значение последнего выражения.

5.3. Функции

Crystal Reports 9 содержит более ста встроенных функций. Рассмотрим некоторые из них.

5.3.1. Функции работы с текстом (String)

Текст, поле типа string или текстовая переменная в формуле может восприниматься как массив, поэтому для выделения символа или последовательности символов необходимо использовать индекс. Так, формула

```
"Иванов"[3]
```

вернет значение а — третий символ слева, а формула

```
"Иванов" [-2]
```

вернет значение о — второй символ справа. Формула

```
{customer.fax}[1 to 3]
```

вернет три символа поля {customer.fax} с первого по третий.

Для выделения отдельного символа или подстроки может также использоваться функция Mid(str, Start, Lenth), где str — строка, Start — первая позиция выделяемой подстроки, Lenth — длина выделяемой подстроки. Функция

```
Mid(customer.fax}, 1, 3)
```

также вернет первые три символа поля {customer.fax}.

Функция Trim(string) удаляет пробелы в строке, функция TrimRigth(string) удаляет только пробелы справа, TrimLeft(string) — слева. Предположим, в полях таблицы Customer содержатся следующие значения: Title — "Проф.",

Contact First Name — "Иван", Contact Last Name — "Иванов". Тогда формула

```
TrimRight ({Customer.Contact Title})+" "
+ TrimRight ({Customer.Contact First Name}) +" "
+ {Customer.Contact Last Name}
```

вернет значение "Проф. Иван Иванов".

Функция ToText(number, dec_place,"thouzand separator", "decimal separator") конвертирует число в текст. Здесь number — число, dec_place — количество знаков после запятой, thouzand separator — разделитель тысяч, decimal separator — разделитель дробной части. Три последних аргумента могут быть опущены. Например, если значение поля {PI} равно 3.1415926, то функция

```
ToText({PI}, 4)
```

вернет значение 3.1415.

Функция ToWords(number, dec_place) возвращает сумму прописью числа number на английском языке. Здесь dec_place — количество знаков после запятой. Так, функция

```
ToWords(3635.235, 4)
```

вернет значение "three thousand six hundred thirty-five and 2350/10000".

Для получения числа прописью на русском языке можно использовать функции из библиотек UFL *(разд. 5.5).*

В старых информационных системах строковые данные часто хранились в верхнем регистре, например, "ИВАНОВ". Чтобы в отчете получить "Иванов", можно использовать функции перевода текста в нижний регистр — LowerCase или верхний регистр — UpperCase, а также функцию определения длины строки Length. Так, если в поле {Customer.Contact First Name} хранится значение "ИВАНОВ", то формула

```
UpperCase({Customer.Contact First Name} [1]) + LowerCase
({Customer.Contact First Name} [2 to Length({Customer.Contact First Name}) ] )
```

вернет значение "Иванов".

Функция ReplicateString("string", n) повторяет значение строки string n раз. Например, функция

```
ReplicateString("*",5)
```

вернет значение *****.

5.3.2. Арифметические функции (Math)

Набор арифметических функций включает тригонометрические функции (Sin, Cos, Tan, Atn), экспоненту (Exp) и логарифм (Log), функцию отбрасывания дробной части (Truncate), функцию-генератор случайных чисел (Rnd) и некоторые другие.

5.3.3. Агрегатные функции (Summary)

В Crystal Reports 9 богатый набор агрегатных функций (группа Summary в списке функций). Каждая агрегатная функция имеет три или более варианта — с одним, двумя и тремя аргументами. Функция с одним аргументом производит подсчет значений по всему отчету. Например, функция

```
Count({Customer.Customer Name})
```

возвращает количество клиентов по всему отчету.

Функция с двумя аргументами производит подсчет в группе. В качестве второго аргумента должно использоваться поле, по которому в отчете уже произведено группирование. Например, функция

```
Count({Customer.Customer Name}, {Customer.Country})
```

возвращает количество клиентов в стране, в отчете должна быть группировка по полю Customer.Country.

Третий аргумент используется в агрегатных функциях, если группирование производится по дате или логическому полю (Boolean). Если данные группируются по дате, то должен быть указан период, по которому производится группировка, например по месяцам, по неделям, по дням и т. д. Тот же признак должен быть указан в качестве третьего параметра агрегатной функции

```
Count({Orders.Order Amount}, {Orders.Order Date}, "monthly").
```

Эта функция вернет количество заказов в каждом месяце. В отчете должна быть группировка по полю Orders.Order Date по месяцам.

Аргументом агрегатной функции может быть массив. Так, функция

```
Sum([100,75,50])
```

вернет значение 225.

Упражнение 5.1

Используя в качестве источника данных таблицу Customer, создайте отчет и включите поля Customer.Last Year's Sales и Customer.Customer Name. Создайте группировку по городам (Customer.City). Вставьте сумму по каждому городу, для этого:

1. Создайте формулу {@ CityGroupSum}, содержащую функцию Sum ({Customer.Last Year's Sales}, {Customer.City}).
2. Вставьте ее в секцию Group Footer.
3. В секцию Details вставьте формулу, рассчитывающую процент продаж клиента относительно общего объема продаж в городе.
4. {Customer.Last Year's Sales}%{@CityGroupSum}.

5. Добавьте в отчет группировку по стране (Customer.Country). В подвал (секция Group Footer) группы по стране добавьте формулу, подсчитывающую количество клиентов в каждой стране:

```
"Количество клиентов в стране " + {Customer. Country} + " - " +
ToText(DistinctCount({Customer.Customer Name}, {Customer.Country}) , 0)
```

5.3.4. Функции для работы с датами и временем (Date/Time)

Функции, объединенные в группу Date/Time, позволяют конвертировать и обрабатывать поля и переменные типа дата. Результат операций с такими полями зависит от типа операции. Так, формула

```
{Orders.Ship Date} - {Orders.Order Date}
```

возвращает число, равное количеству дней от дня заказа — поле типа дата Orders. Order Date — до дня отгрузки — Orders.Ship Date.

Функции Year({Orders.Order Date}), Month({Orders.Order Date}), Day({Orders.Order Date}) возвращают число — год, месяц и число даты заказа соответственно. Функция DayOfWeek({Orders.Ship Date}) возвращает номер дня недели (по умолчанию первым днем недели считается воскресенье) и может быть использована в формуле, возвращающей день недели прописью:

```
["Воскресенье","Понедельник",
"Вторник","Среда","Четверг","Пятница","Суббота"]
[DayOfWeek({Orders.Ship Date})]
```

В этой формуле в первых квадратных скобках задается массив строк (["Воскресенье","Понедельник"]). Во вторых квадратных скобках вычисляется индекс массива. Формула возвращает строку элемент массива, соответствующую вычисленному индексу. Аналогично, с использованием функции Month(), можно написать формулу, возвращающую месяц прописью.

Функция Date(YYYY, MM, DD) имеет три числовых аргумента — год, месяц и день и возвращает дату. Эту функцию можно использовать для конвертации строки в дату при нестандартном формате хранения даты, например, YYYY/MM/DD:

```
Date(ToNumber({StringDate})[1 to 4], ToNumber({StringDate})[6 to 7],
ToNumber({StringDate})[9 to 10])
```

Функция DateTimeValue возвращает дату-время и аналогична Date. Эта функция может быть использована для конвертации значения строки или числа в дату. Допустимые значения аргументов DateTimeValue приведены в табл. 3.1.

Таблица 5.3. Допустимые значения аргументов функции DateTimeValue

Функция	Описание аргумента
DateTimeValue (date)	Date — значения типа дата
DateTimeValue (date, time)	Date — значения типа дата , time — значения типа время
DateTimeValue (number)	Number — количество дней от 30 декабря 1899 года, например, DateTimeValue(20) вернет дату 19 января 1900 года
DateTimeValue (string)	String — строка, представляющая дату и время, например "September 15, 2002, 10:45a.m."
DateTimeValue (year, month, day)	year — год, например: 1996 month — месяц, например 1 соответствует январю day — день месяца
DateTimeValue (year, month, day, hour, min, sec)	year — год, например: 1996 month — месяц, например 1 соответствует январю day — день месяца hour — часы min — минуты sec — число, представляющее секунды

5.3.5. Дополнительные функции (Additional Functions)

В разделе дополнительных функций (Additional Functions) содержатся функции преобразования дат: DTSToDateTime({datetimestring}), DTSToDateTime({datetimestring}) и DTSToTime({datetimestring}). Эти функции предназначены для конвертации строки в дату или дату-время. Так, функция

```
DTSToDateTime ("2000/01/13 11:30:15")
```

вернет значение типа дата-время 2000/01/13 11:30:15.

Функция DateTimeToDate(DateTime) конвертирует поле типа дата-время в дату, DateTimeToSecond(DateTime) возвращает количество секунд после полуночи, а DateTimeToTime(DateTime) — время в "военном" формате 00:00:00.

Дополнительные функции содержатся в библиотеках функций, определяемых пользователем (UFL). В списке функций окна **Formula Editor** они сгруппированы по библиотекам. В наименовании каждой группы указывается наименование файла библиотеки в формате u2*.dll *(разд. 5.5)*.

5.3.6. Функции периода времени (Date Ranges)

Функции периода времени (Date Ranges) не имеют аргументов и возвращают диапазон дат, который может быть использован для сравнения. В табл. 5.4 приведены периоды времени, возвращаемые этими функциями.

Таблица 5.4. *Функции периода времени*

Функция	Описание
Aged0To30Days	Все дни с сегодняшнего и до 30 дня тому назад
Aged31To60Days	То же с 31 по 60 день
Aged61To90Days	То же с 61 по 90 день
Over90Days	Более 90 дней тому назад
AllDatesFromToday	Все будущие дни после сегодняшнего
AllDatesFromTomorrow	Все будущие дни после завтрашнего
AllDatesToToday	Все дни, предшествующие сегодняшнему
AllDatesToYesterday	Все дни, предшествующие вчерашнему
Calendar1stHalf	Все дни первой половины календарного года
Calendar2ndHalf	Все дни второй половины календарного года
Calendar1stQtr	1 квартал
Calendar2ndQtr	2 квартал
Calendar3rdQtr	3 квартал
Calendar4thQtr	4 квартал
Last4WeeksToSun	4 недели, предшествующие последнему воскресенью. Начинаются в понедельник и кончаются в воскресенье. Например, если сегодня воскресенье, то период исчисляется с понедельника, бывшего четыре недели назад, по сей день включительно. Если сегодня 22 сентября 1996 года, Last4WeeksToSun начинается 26 августа (понедельник) и кончается сегодня, 22 сентября (воскресенье). Если сегодня 28 сентября 1996 года, то Last4WeeksToSun начинается 26 августа (понедельник) и кончается 22 сентября (предыдущее воскресенье).
Last7Days	Последние семь дней, предшествующие сегодняшнему
LastFullMonth	С первого по последний день предыдущего месяца
LastFullWeek	С воскресенья по субботу прошлой недели
LastYearMTD	Все дни текущего месяца прошлого года

Таблица 5.4 (окончание)

Функция	Описание
LastYearYTD	Все дни прошлого года, вплоть до текущей даты прошлого года
MonthToDate	С первого дня месяца по сегодняшний день
Next...Days	Даты за период, начинающийся с сегодняшнего дня
WeekToDateFromSun	С последнего воскресенья по сегодняшний день
YearToDate	С первого дня календарного года по сегодняшний день

Диапазоны дат рассчитываются относительно даты печати, что соответствует специальному полю **PrintDate**. По умолчанию в качестве даты печати берется системная дата. Однако эту дату можно изменить в диалоговом окне **Set Print Date and Time** (рис. 5.4), которое можно вызвать, выбрав пункт меню **Report/Set Print Date and Time**.

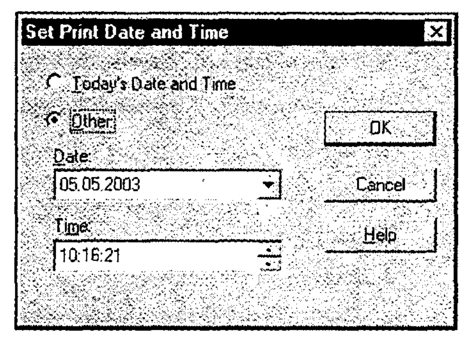


Рис. 5.4. Диалоговое окно **Set Print Date and Time**

Упражнение 5.2

Используя в качестве источника данных таблицу **Orders**, создайте отчет и включите поля **Orders.Ship Date** и **Orders.Order Amount**. Подсчитайте относительную долю заказов, отгруженных от текущего дня до 30 дней тому назад, от 31 до 60 дней тому назад, от 61 до 90 дней тому назад и более 90 дней тому назад. Для этого в секцию **Details** внесите 4 формулы и скройте их. Для того чтобы скрыть формулу, необходимо щелкнуть по ней правой кнопкой мыши, в контекстном меню выполнить команду **Format Field** и в диалоге **Format Editor** включить опцию **Suppress**.

1. 0 to 30:

```
If {Orders.Ship Date} in Aged0To30Days then {Orders.Order Amount} else 0;
```

2. 31 to 60:

```
If {Orders.Ship Date} in Aged31To60Days then {Orders.Order Amount} else 0;
```

3. 61 to 90:

```
If {Orders.Ship Date} in Aged61To90Days then {Orders.Order Amount} else 0;
```

4. 90 Plus:

```
If {Orders.Ship Date} in Over90Days then {Orders.Order Amount} else 0.
```

В секцию **Report Footer** включите следующие формулы.

1. Total for Report — вычисляет общее количество заказов:

```
Sum ({@0 to 30})+Sum ({@31 to 60})+Sum ({@61 to 90})+Sum ({@90 Plus});
```

2. Percent 0 to 30 — вычисляет относительную долю заказов, отгруженных от текущего дня до 30 дней тому назад:

```
Sum ({@0 to 30}) % {@Total for Report};
```

3. Percent 31 to 60 — вычисляет относительную долю заказов, отгруженных от 31 до 60 дней тому назад:

```
Sum ({@31 to 60}) % {@Total for Report};
```

4. Percent 61 to 90 — вычисляет относительную долю заказов, отгруженных от 61 до 90 дней тому назад:

```
Sum ({@61 to 90}) % {@Total for Report};
```

5. Percent 90 Plus — вычисляет относительную долю заказов, отгруженных более 90 дней тому назад:

```
Sum ({@90 Plus}) % {@Total for Report}.
```

Измените дату отчета на 1.01.2001 и посмотрите результат на последней странице отчета.

5.3.7. Функции состояния печати Print State

В разделе Print State содержится ряд полезных функций, позволяющих управлять печатью отчета. Функция OnFirstField возвращает булево значение "истина", если строка печатается впервые. Функция OnLast Records возвращает значение "истина", если печатается последняя строка отчета. Функция InRepeatedGroupHeader возвращает значение "истина", если заголовок группы печатается не впервые. Функции Next(fld) и Previos(fld) возвращают значение поля fld из предыдущей и последующей строк соответственно. Функции IsNull(fld), NextIsNull(fld), PreviosIsNull(fld) возвращают значение "истина", если значение поля fld текущей, последующей или предыдущей строки соответственно принимает значение NULL.

Функция PageNumber возвращает номер страницы, TotalPageCount — общее количество страниц в отчете, функция PageNofM — строку

Page N of M,

где *N* — номер текущей страницы, *M* — общее количество страниц в отчете.

5.3.8. Функции свойств документа (Document Properties)

Функции свойств документа (Document Properties) возвращают значения свойств отчета:

- дату и время печати отчета — PrintDate и PrintTime;
- дату и время последнего изменения отчета — ModificationDate и ModificationTime;
- дату и время последнего обновления данных отчета — DataDate и DataTime;
- заголовок и комментарий к отчету — ReportTitle и ReportComments;
- наименование, дату создания и имя автора файла отчета — Filename, FileCreationDate, FileAuthor.

5.3.9. Функции времени выполнения (Evaluation Time)

Crystal Reports 9 является многопроходным генератором отчетов, другими словами, он генерирует отчет в несколько этапов. На каждом этапе выполняются вычисления по формулам определенного типа. Существует пять этапов выполнения отчета.

Предварительный первый этап — BeforeReadingRecords. На этом этапе производятся вычисления по формулам, не содержащим специальные поля, поля базы данных или суммирующие поля, например формула 2*2.

Основной первый этап — WhileReadingRecords. Выполняется чтение полей из базы данных, вычисление по формулам, содержащим поля базы данных, вычисление по формулам выборки данных, сортировка, группировка, создание матричных таблиц.

Предварительный второй этап. Выполняется группирование Top N и иерархическая группировка.

Основной второй этап — WhilePrintingRecords. Выполняется вычисление по формулам, содержащим суммирующие поля и функции, поля running totals, специальные поля типа номера страницы и функции состояния печати.

Третий этап. Вычисляются формулы, содержащие специальные поля и функции, возвращающие общее количество строк в отчете (например, TotalPageCount или PageNofM).

Существует возможность переноса выполнения формулы на более поздний этап, но не на более ранний. Для этой цели Crystal Reports 9 содержит функции времени выполнения:

BeforeReadingRecords — указывает, что формула должна выполняться на этапе BeforeReadingRecords, перед чтением данных;

WhileReadingRecords — указывает, что формула должна выполняться на этапе WhileReadingRecords, во время чтения данных;

WhilePrintingRecords — указывает, что формула должна выполняться на этапе WhilePrintingRecords, во время печати;

EvaluateAfter({@formula}) — указывает, что формула должна выполняться после выполнения формулы, указанной в качестве аргумента {@formula}, например, EvaluateAfter({@SetVariable}).

5.3.10. Функции работы с массивами

Crystal Reports 9 имеет набор функций обработки массивов и поддерживает динамические массивы. Функция MakeArray() позволяет задать значения массива. С той же целью можно использовать квадратные скобки. Так, выражения, определяющие переменные типа массив

```
stringVar array x:= MakeArray ("a", "bb", "ccc") ;
```

и

```
stringVar array x:= ["a", "bb", "ccc"] ;
```

эквивалентны.

Для применения динамических массивов можно использовать функции Redim и Redim Preserve.

Функция Redim — переопределяет размеры массива с потерей всех предварительно определенных значений, например, после выполнения следующей формулы

```
stringVar array x:= MakeArray ("a", "bb", "ccc") ;
Redim x[4] ;
x[4] = "dddd"
```

значения массива x будут равны ("", "", "", "dddd"), а после выполнения

```
stringVar array x:= MakeArray ("a", "bb", "ccc") ;
Redim Preserve x[4] ;
x[4] = "dddd"
```

массив x примет значения ("a", "bb", "ccc’’, "dddd").

Функция UBound(Array) возвращает размер массива. Например, формула

```
Local NumberVar Array simpleArray;
Redim simpleArray[10];
UBound(simpleArray)
```

возвращает значение 10.

5.3.11. Специальные функции (Custom functions)

Специальные функции представляют собой создаваемые разработчиком процедуры, возвращающие значение. Однажды созданная специальная функция может многократно использоваться как в отчете, в котором она создана, так и в других отчетах, если она помещена в хранилище. Использование специальных функций позволяет поддерживать единую логику при разработке всех отчетов в рамках компании.

Специальные функции могут быть созданы как в синтаксисе Crystal, так и Basic. Специальная функция сначала должна быть создана в среде **Custom Function Editor**, а затем включена в формулу с помощью **Formula Expert**. Отметим, что инструменты создания и использования специальных функций входят во все варианты поставки Crystal Reports 9, кроме варианта Standard.

Для создания специальной функции необходимо в диалоговом окне **Formula Workshop** раскрыть список создания нового объекта (табл. 5.1) и выбрать пункт **Custom Function**. Появляется диалоговое окно **Custom Function Name** (рис. 5.5).

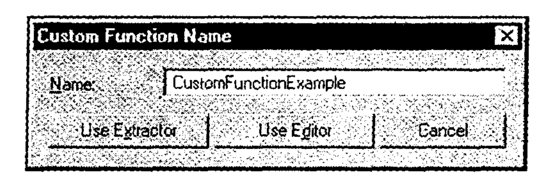


Рис. 5.5. Диалоговое окно **Custom Function Name**

В диалоговом окне **Custom Function Name** следует ввести имя специальной функции. Для создания функции на основе существующей формулы следует щелкнуть на кнопке **Use Extractor**, для создания функции "с нуля" — на кнопке **Use Editor.** Кнопка **Use Editor** вызывает окно **Custom Function Editor**, интерфейс которого подобен окну **Formula Editor** *(разд. 5.1)*, однако в отличие от него не имеет списка объектов отчета, поскольку синтаксис специальных функций запрещает использование объектов отчета в качестве формальных аргументов.

Создание специальной функции в среде **Custom Function Editor** (рис. 5.6) не отличается от создания формулы в **Formula Editor**. Двойной щелчок на списке операторов или формул включает оператор или формулу в текст специальной функции, который показывается в нижнем окне.

Специальная функция должна начинаться служебным словом Function, содержать список аргументов и исполняемый код, например

```
Function (NumberVar valueToIncrease, numberVar percentIncrease)
valueToIncrease + valueToIncrease * percentIncrease / 100;
```

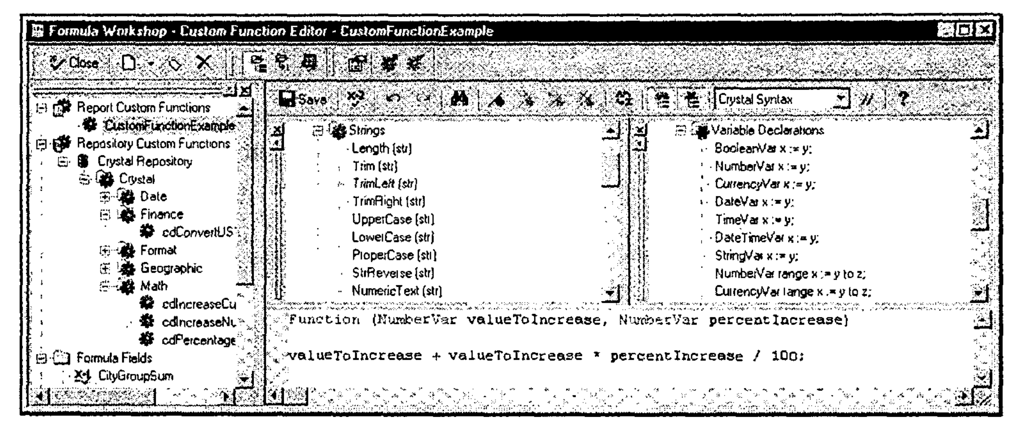


Рис. 5.6. Диалоговое окно **Custom Function Editor**

Здесь "NumberVar valueToIncrease, numberVar percentIncrease" — список аргументов, "valueToIncrease + valueToIncrease * percentIncrease / 100;" — исполняемый код. Исполняемый код может быть заключен в круглые скобки.

Список аргументов разделяется запятой, допускается пустой список. Служебное слово Optional означает, что при вызове функции аргумент может быть опущен. Для аргументов, описанных как Optional, требуется задание значения по умолчанию, например

```
Optional NumberVar X:=0
```

Допускается использование аргументов следующих типов:

- NumberVar — число;
- CurrencyVar — валюта;
- BooleanVar — логический (Boolean);
- DateVar — дата;
- DateTimeVar — дата-время;
- TimeVar — время;
- StringVar — строка.

Возможно использование в качестве аргументов массивов и диапазонов, кроме диапазона типа Boolean.

Исполняемый код должен вычислять возвращаемое значение функции. Синтаксис специальных функций накладывает следующие ограничения на исполняемый код:

- нельзя использовать поля отчета, включая агрегатные, и поля базы данных;
- допускается использование только локальных переменных, т. е. нельзя использовать переменные, описанные как use shared или global;

- запрещается использование рекурсии;
- запрещается использование библиотек UFL *(разд. 5.4)*;
- запрещается использование функций времени выполнения (Evaluation Time), функций состояния печати (Print State), функций свойств документа (Document Properties), а также функций Rnd, CurrentFieldValue, DefaultAttribute и GridRowColumnValue.

После создания специальной функции следует определить ее свойства. Для этого необходимо щелкнуть на кнопке, осуществляющей переключение между окнами **Custom Function Editor** и **Custom Function Properties** (табл. 5.1), и в окне **Custom Function Properties** (рис. 5.7) внести описание (**Summary**), категорию (**Category**) и автора (**Author**) специальной функции. Наименование категории будет использовано как папка в древовидном списке **Formula Workshop** в разделе специальных функций.

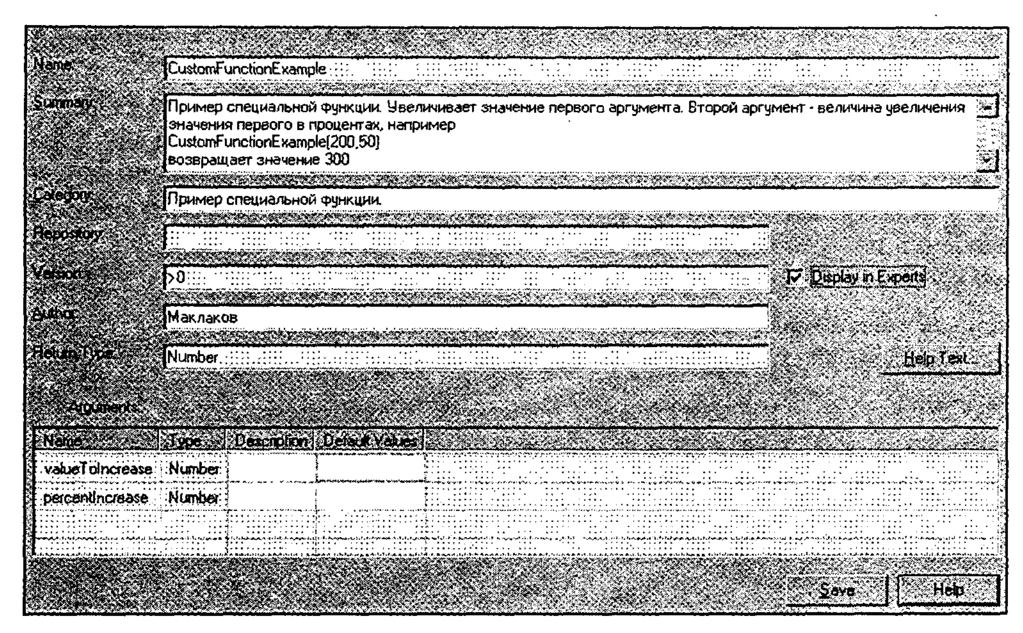


Рис. 5.7. Окно **Custom Function Properties**

Нередактируемое поле **Repository** показывает наименование хранилища, в которое помещена функция. В поле **Version** отображается автоматически генерируемый номер версии функции. Номер версии обновляется каждый раз при обновлении специальной функции в хранилище. Поле **Return Type** показывает тип возвращаемого значения и зависит от типа аргументов.

В списке **Arguments** для каждого аргумента можно задать описание и значение по умолчанию.

Новую специальную функцию можно создать на основе существующей формулы. Для этого в диалоговом окне **Custom Function Name** следует щелкнуть на кнопке **Use Extractor**. Появляется диалоговое окно **Extract Custom Function from Formula** (рис. 5.8).

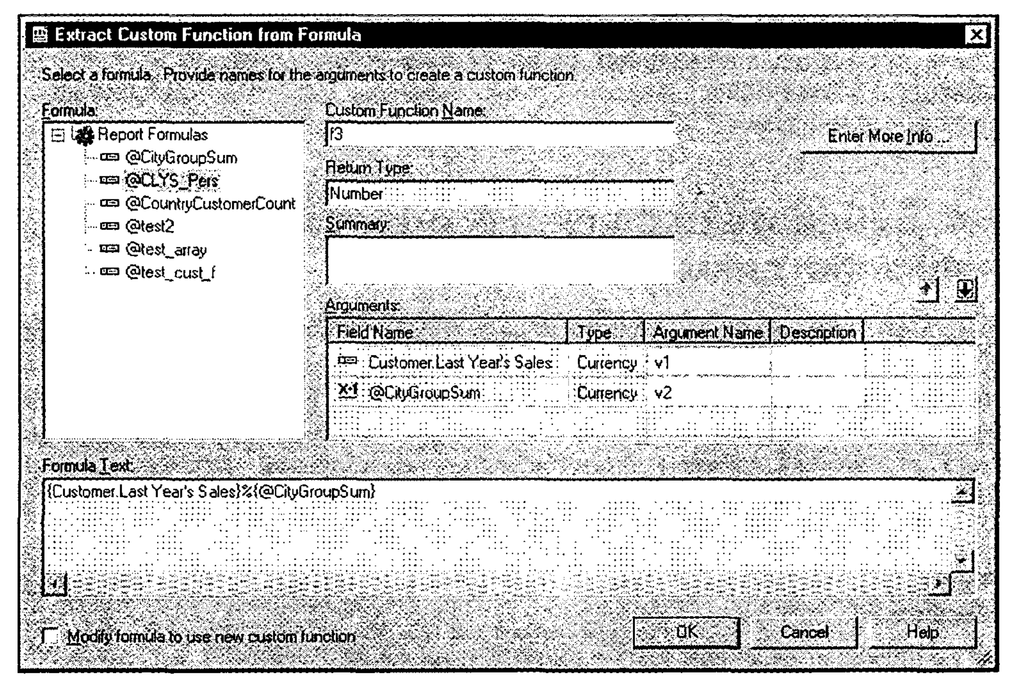


Рис. 5.8. Диалоговое окно **Extract Custom Function from Formula**

В поле **Formula Text** отображается текст формулы, на основе которой будет создана специальная функция. Поля отчета и базы данных заменяются аргументами. Тип аргумента соответствует типу поля отчета, имя аргумента по умолчанию генерируется как vN, где N — порядковый номер, например v1,v2 v3 и т. д. Например, на основе формулы

```
{Customer.Last Year's Sales}%{@CityGroupSum}
```

будет создана специальная функция

```
Function (currencyVar v1, currencyVar v2)
v1%v2
```

В полях **Custom Function Name** и **Summary** можно задать имя и описание специальной функции.

Для размещения специальной функции в хранилище необходимо щелкнуть по кнопке перемещения функции в хранилище в палитре инструментов

Formula Workshop (табл. 5.1). Открывается диалоговое окно **Opens the Add Custom Function to Repository**, в котором следует щелкнуть на кнопке **OK**.

Для включения специальной функции в отчет из хранилища следует переместить указатель на соответствующую строку в древовидном списке **Formula Workshop** и щелкнуть на кнопке внесения функции в отчет (табл. 5.1).

После включения специальной функции в отчет она появляется в списке функций **Formula Editor** в разделе **Custom Functions** и ее можно использовать при создании формул так же, как и встроенные функции Crystal Reports 9.

5.4. Библиотеки функций, определяемых пользователем (UFL)

Функции, определяемые пользователем (UFL), по назначению подобны специальным функциям, однако в отличие от них создаются во внешних средах программирования — C++, Delphi и Visual Basic. Код библиотеки функций UFL должен быть откомпилирован в динамическую библиотеку, файл библиотеки должен быть размещен в корневом каталоге Crystal Reports 9, например

C:\ Program Files\Crystal Decisions\Crystal Reports 9.

Имя файла библиотеки должно начинаться с символов u2l, например u2lsample.dll.

Ниже приведен фрагмент кода библиотеки на языке C++.

```
#include <Windows.h>
#include "UFDll.h"
#include "UFMain.h"
#include "UFUser.h"
#define PicturePlaceholder 'x'
UFFunctionDefStrings FunctionDefStrings [] =
{
{"String Picture (String, String)",
 Picture},
{NULL, NULL, NULL}
};
```

Код должен "подключать" модули UFDll.h, UFMain.h и UFUser.h, которые входят в поставку Crystal Reports 9.

Новые функции появляются в списке функций в разделе Additional Functions после перезапуска Crystal Reports 9.

5.5. Переменные

Использование переменных в формулах включает три стадии:

1. Декларирование переменной.
2. Присвоение значений.
3. Использование для вычислений.

При декларировании переменной запрещено использование в качестве имени зарезервированных слов и имен формул, нельзя также использовать имена длиной более 256 символов. Переменная может быть объявлена как локальная (Local), глобальная (Global) и разделяемая (Shared). Локальная переменная сохраняет значение только в пределах формулы, в которой она определена. Глобальная переменная сохраняет значение в пределах отчета. Разделяемая переменная сохраняет значение в пределах отчета, включая подотчеты, создание отчетов с подотчетами будет описано в *главе 6*. При использовании переменной более чем в одной формуле, она должна быть декларирована в каждой из них как глобальная или разделяемая. По умолчанию, если рамки применения переменной не указаны, она воспринимается как глобальная.

Операторы декларирования переменных находятся в нижней части списка операторов в правом окне редактора Formula Editor.

Поддерживаются несколько типов переменных. В первую группу типов входят те же типы, что и типы полей базы данных:

- NumberVar — число;
- CurrencyVar — валюта;
- BooleanVar — логический (Boolean);
- DateVar — дата;
- DateTimeVar — дата-время;
- TimeVar — время;
- StringVar — строка.

Декларирование переменной может быть совмещено с присвоением значений. В этом случае должен применяться оператор присваивания :=. В правой части оператора присвоения может быть не только константа, но и арифметическое выражение. Ниже приведены четыре примера декларирования переменной:

```
NumberVar Amount;
NumberVar Quantity :=5;
CurrencyVar SellPrice:={Product.Price}*1.1;
BooleanVar OverQuota := {Sales} > {Quota};.
```

Во вторую группу входят переменные диапазона:

- NumberVar range — диапазон чисел;
- CurrencyVar range — диапазон валют;
- DateTimeVar range — диапазон дат-времени;
- DateVar range — диапазон дат;
- TimeVar range — диапазон времени;
- StringVar range — диапазон строк.

Переменные диапазона могут использоваться в операторах сравнения. Ниже приведен пример формулы, использующей переменную диапазона:

```
NumberVar Range GR := 20 to 30;
if {Orders Detail.Unit Price} in GR then 1 else 0.
```

В третью группу входят массивы:

- NumberVar array — массив чисел;
- CurrencyVar array — массив валют;
- DateTimeVar array — массив дат-времени;
- DateVar array — массив дат;
- TimeVar array — массив времени;
- StringVar array — массив строк.

Примеры декларирования массива:

```
StringVar array Weekdays:= ["Воскресенье" ,"Понедельник", ...]
Local NumberVar Array x := MakeArray (1, 1000, 10, 100, 10);
```

В последнюю группу входят массивы диапазона:

- NumberVar range array x:=[a to c, y to z] — массив диапазона чисел;
- CurrencyVar range array — массив диапазона валют;
- DateTimeVar range array — массив диапазона дат-времени;
- DateVar range array — массив диапазона дат;
- TimeVar range array — массив диапазона времени;
- StringVar range array — массив диапазона строк.

Пример декларирования массива диапазона:

Shared StringVar Range Array y := ["A" To "C", "H" To "J"];

Упражнение 5.3

Вычисление кумулятивного (накапливающего) значения.

Используя в качестве источника данных таблицу **Orders**, создайте новый отчет и включите поле **Orders.Order Amount**. Создайте формулу, значение

которой равнялось бы сумме значений поля **Orders.Order Amount** из всех предыдущих строк. Для этого

в секцию **Report Header** поместите скрытую формулу **Reset**:

```
currencyVar Rvar := 0;
```

в секцию Details поместите формулу **Display**:

```
currencyVar Rvar;
 Rvar:= Rvar + {Orders.Order Amount}
```

Посмотрите результат. Отчет должен выглядеть примерно так:

Order Amount	Display
50$	50$
44$	93$
44$	137$
33$	170$
33$	203$
50$	252$
42$	294$
983$	1 277$

Обратите внимание на то, что по умолчанию отображаются целые значения сумм. Изменить формат отображения суммы можно в диалоговом окне **Format Editor**, которое вызывается из контекстного меню. Выполните команду меню **Report | Record Sort Expert** и установите сортировку по полю **Orders.Order ID**. После этого отчет будет выглядеть так:

Order Amount	Display
3 480$	94 846$
3 480$	98 326$
29$	98 355$
2 447$	104 282$
150$	104 431$
178$	104 609$
3 545$	111 754$

Формула **Display** больше не показывает кумулятивное значение. Это связано с тем, что вычисление формул **Display** и **Reset** происходит на более раннем этапе, чем сортировка данных. Чтобы формула **Display** показывала правильный результат, необходимо добавить в формулы функцию **WhilePrintingRecords**.

Формула **Reset** должна иметь следующий вид:

```
WhilePrintingRecords;
currencyVar Rvar := 0;
```

а формула Display

```
WhilePrintingRecords;
currencyVar Rvar;
 Rvar:= Rvar + {Orders.Order Amount}
```

После этого кумулятивное значение отображается корректно.

5.6. Управляющие операторы

Помимо операторов декларирования переменных, формулы Crystal Reports 9 позволяют использовать операторы следующих типов:

- арифметические операторы: сложение, вычитание, умножение, деление и др.;
- операторы сравнения больше, меньше и др.;
- строковые операторы: конкатенация, выделение подстроки;
- операторы диапазонов: создание диапазона, проверка на вхождение в диапазон;
- булевы операторы: логические "и", "или" и др.;
- операторы работы с массивами: создание массива, изменение размерности массива и др.;
- управляющие операторы: условия, цикла и др.;
- прочие операторы: присвоение, комментарий и др.

Полный список операторов доступен в правом окне редактора **Formula Editor**. Для ввода оператора в текст формулы следует щелкнуть на нем дважды. Рассмотрим более подробно управляющие операторы.

Оператор **if then else** — условный оператор — позволяет выполнить фрагмент кода в зависимости от условия. Формальный синтаксис условного оператора

if X then Y else Z,

где **X** — логическое выражение, возвращающее значение "истина" или "ложь", **Y** и **Z** — выражения, возвращающие значения одного типа, например, строковую или числовую константу. После **else** может следовать другой оператор **if then else**.

Упражнение 5.4

Используя в качестве источника данных таблицу **Customer**, создайте отчет и включите поля **Customer.Last Year's Sales** и **Customer.Customer Name**. Создайте формулу, возвращающую значение "Хорошо", если значение поля **Customer.Last Year's Sales** больше 30000, и "Плохо", если значение поля **Customer.Last Year's Sales** меньше 1000. Разместите ее в секции **Details**.

```
if {Customer.Last Year's Sales}>3000 then " Хорошо " else if
{Customer.Last Year's Sales}<5000 then " Плохо "
```

В некоторых случаях функция **IIF(X,Y,Z)** может заменить оператор **if then else**. Первый аргумент функции X — логическое выражение, второй аргумент Y — выражение, выполняющееся, если X возвращает значение "истина", и третий аргумент Z — выражение, выполняющееся, если X возвращает значение "ложь".

Упражнение 5.5

Используя в качестве источника данных таблицу **Customer**, создайте отчет и включите поля **Customer.Last Year's Sales**, **Customer.Country** и **Customer. Customer Name**. Необходимо подсчитать сумму продаж в прошлом году в Европе, Северной Америке и Южной Америке. Проблема заключается в том, что в базе данных нет признака континента. Один из способов расчета суммы по признаку, который не хранится в базе данных, был рассмотрен в *разд. 3.2* (специальная группировка). Ниже рассматривается другой способ — без группировки, но с использованием формул.

В секции **Details** отчета разместите три формулы:

NA:

```
if {Customer.Country} in ["Canada","USA","Mexico"] then {Customer.Last
Year's Sales} else 0
```

SA:

```
if {Customer.Country} in
["Argentina","Bahamas","Barbados","Brazil","Bermuda","Bolivia","Chile",
"Colombia","Ecuador","Panama","Peru","Venezuela"]
then {Customer.Last Year's Sales} else 0
```

EU:

```
if {Customer.Country} in
["Austria","Belgium","Czech Republic","Denmark",
"England","Finland","France","Germany","Greece","Hungary",
"Ireland","Luxembourg","Netherlands","Northern Ireland",
"Norway","Poland","Portugal","Romania","Russia","Scotland",
"Spain","Sweden","Switzerland","Wales","Ukraine" ,"Italy","UK"]
then {Customer.Last Year's Sales} else 0
```

Создайте суммирующее по отчету поле (**Grand Total**) для каждой формулы **NA**, **SA**, **EU**.

В секции **Report Footer** появятся поля, показывающие общее количество продаж по каждому континенту.

Оператор **select case** подобен условному оператору, но позволяет записать сложные выражения в более компактной форме. Формальный синтаксис оператора **select case** следующий:

```
Select {поле}_case X1 : Y1
case X2 : Y2
.......
 default: YN
```

где {поле} — поле отчета, переменная или параметр, Xn — значения этого поля, Yn — выражения, возвращающие значения одного типа. Например, формула:

```
Select {Customer.Fax}[1 To 3]
Case "604", "250" :
"BC"
Case "206", "509", "360" :
"WA"
Default : "";
```

возвращает значение BC, если первые три цифры поля **Customer.Fax** равны 604 или 250, "WA" — если 206, 509 или 360, и пустую строку во всех прочих случаях.

Операторы цикла позволяют организовать вычисления в цикле. Crystal Reports 9 содержит три оператора цикла:

- **for to step do**;
- **while do**;
- **do while**;

и два оператора выхода из цикла:

- **exit for**;
- **exit while**.

Формальный синтаксис оператора **for to step do**:

```
for i:=N to M step K do X
```

где

N — начальное значение счетчика цикла,

M — конечное значение счетчика цикла,

K — шаг счетчика цикла,

X — операторы, выполняемые в цикле.

Ниже показана формула, обращающая строку так, что первый символ становится последним, второй — предпоследним и т. д.

```
//Обращение строки
Local StringVar str := "";
Local NumberVar strLen :=
 Length ({Customer.Customer Name});
Local NumberVar i;
For i := strLen To 1 Step -1 Do
(
   str := str + {Customer.Customer Name}[i]
);
str
```

Оператор цикла **While** имеет два варианта использования:

- **While X Do Y**;
- **Do Y While X**,

где X — логическое условие, Y — операторы, выполняемые в цикле.

Операторы выхода из цикла **exit for** и **exit while** могут входить в число операторов, выполняемых в цикле, и позволяют прекратить выполнение оператора цикла.

Например, если массив N имеет четыре значения

["Frank", "Helen", "Fred", "Linda"], то следующая формула возвращает значение 3.

```
StringVar Array N;
Local NumberVar i;
Local NumberVar result := -1;
For i :=1 to 4 Do
(
If N[i] = "Fred" Then
(result := i;
Exit For)
);
result
```

5.7. Использование формулы для выборки данных

Формулы Crystal Reports 9 могут быть использованы для вычислений, для выборки, форматирования объектов отчета по условию *(разд. 6.1)*, а также для создания специальных сообщений **Alerts**.

В *главе 3* было показано, как с помощью редактора **Select Expert** можно задать условия выборки. При этом Crystal Reports 9 создает формулу, возвращающую логическое значение. Просмотреть текст формулы можно, щелкнув по кнопке **Show Formula** в диалоговом окне **Select Expert** (рис. 3.1). Формулы, созданные автоматически при помощи **Select Expert**, имеют ограниченные возможности. В частности, если создается несколько условий выборки, то они объединяются логическим "и". Чтобы создать более сложные условия выборки, следует воспользоваться диалоговым окном **Record Selection Formula Editor**, которое можно вызвать из **Select Expert**, последовательно щелкнув на кнопках **Show Formula** и **Formula Editor**. Интерфейс диалогового окна **Record Selection Formula Editor** идентичен интерфейсу **Formula Editor**, рассмотренному ранее. Заметим, что если формула, созданная с помощью **Record Selection Formula Editor**, более сложна, чем та, которую можно получить с помощью **Select Expert** (например, если формула содержит оператор "OR"), то редактирование условий выборки на вкладках **Select Expert** становится невозможным.

С помощью **Record Selection Formula Editor** создаются формулы, которые применяются к каждой строке отчета. Однако, когда отчет содержит группировку данных и вычисление агрегированных данных по группам, возникает необходимость отбора групп, например групп с наибольшим или наименьшим значением агрегированных данных, групп с определенным названием и т. д. Для такой выборки служит специальный инструмент — **Group Selection Formula Editor**. Для вызова **Group Selection Formula Editor** следует в диалоговом окне **Select Expert**, щелкнув на кнопках **Show Formula**, выбрать опцию Group Selection и затем щелкнуть на кнопке **Formula Editor**. Появляется окно **Group Selection Formula Editor**, и в нем в списке полей отчета доступны имена групп и агрегатных полей (Summary).

Формулы для выборки данных не имеют имени. В каждом отчете может быть только одна формула для выборки записей и одна формула для выборки групп.

Отметим, что текст формул для выборки может быть размещен в отчете. Для этого следует включить в отчет специальные поля **Record Selection Formula** и **Group Selection Formula**.

5.8. Специальные сообщения *Alerts*

Специальные сообщения **Alerts** создаются разработчиком отчета для того, чтобы на этапе просмотра отчета при выполнении заранее заданного условия возникало диалоговое окно с сообщением. Специальные сообщения могут быть полезны при обработке сложной информации, когда следует привлечь внимание пользователя к каким-либо необычным или важным фактам, например, когда поля или формулы принимают аномально большое или малое значение. Для специального сообщения необходимо создать

формулу, возвращающую значение "истина" или "ложь". Если при выполнении отчета на этапе **WhilePrintingRecords** формула принимает значение "истина", возникает диалог с сообщением. Сообщением может быть текст или формула, т. е. сообщение может генерироваться динамически из текста и значений полей отчета. При выполнении отчета диалоговое окно с сообщением возникает столько раз, сколько раз формула принимает значение "истина".

Для создания специального сообщения следует выполнить команду меню **Report | Alerts | Create or Modify Alerts**. Появляется диалоговое окно **Create Alerts** (рис. 5.9), в котором следует щелкнуть на кнопке **New**. Для редактирования уже существующего сообщения следует переместить указатель на строку в списке сообщений и щелкнуть на кнопке **Edit**.

Возникает диалоговое окно **Edit Alert** (рис. 5.10). В поле **Name** следует задать имя специального сообщения, в поле **Message** — текст, который будет отображаться в диалоге при выполнении условия.

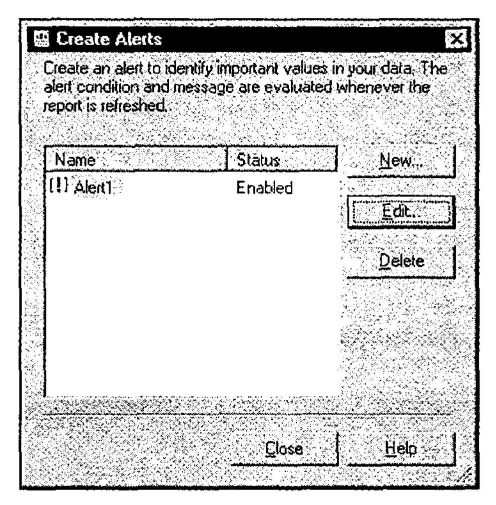


Рис. 5.9. Диалоговое окно **Create Alerts**

Edit Alert
Name Alert1
Message
Condition... Enabled
OK Cancel Help

Рис. 5.10. Диалоговое окно **Edit Alert**

Если текст сообщения должен генерироваться динамически, следует щелкнуть на кнопке справа от поля **Message**. Возникает диалоговое окно **Formula Workshop — Alert Message Formula Editor**, в котором можно создать формулу, возвращающую текстовое сообщение, например,

```
"Продажи компании "+{Customer.Customer Name}+" в прошлом году превысили
$100 000 и составили $"+ToText ({Customer.Last Year's Sales},0 )
```

В формуле можно использовать функцию DefaultAttribute, которая будет принимать значение поля **Message** диалогового окна **Edit Alert**. Например, если значение поля Message — "Продажи компании ", то формула

```
DefaultAttribute +{Customer.Customer Name}+" в прошлом году превысили
$100 000 и составили $"+ToText ({Customer.Last Year's Sales},0 )
```

будет возвращать то же значение, что и приведенная выше.

Для создания условия специального сообщения следует щелкнуть на кнопке **Condition**. Возникает диалоговое окно **Formula Workshop — Alert Condition Formula Editor**, в котором следует создать формулу, возвращающую значение "истина" или "ложь", например,

```
{Customer.Last Year's Sales}>100000
```

Формула условия может включать поля отчета, суммирующие поля, но не должна содержать кумулятивных полей (running totals) и функций состояния печати.

При переходе в режим просмотра отчета или при его обновлении в случае выполнения условия специального сообщения возникает диалоговое окно **Report Alerts** (рис. 5.11), в котором отображается текст сообщения. Щелчок на кнопке **View Records** устанавливает указатель на строку отчета, на которой формула условия возвратила значение "истина".

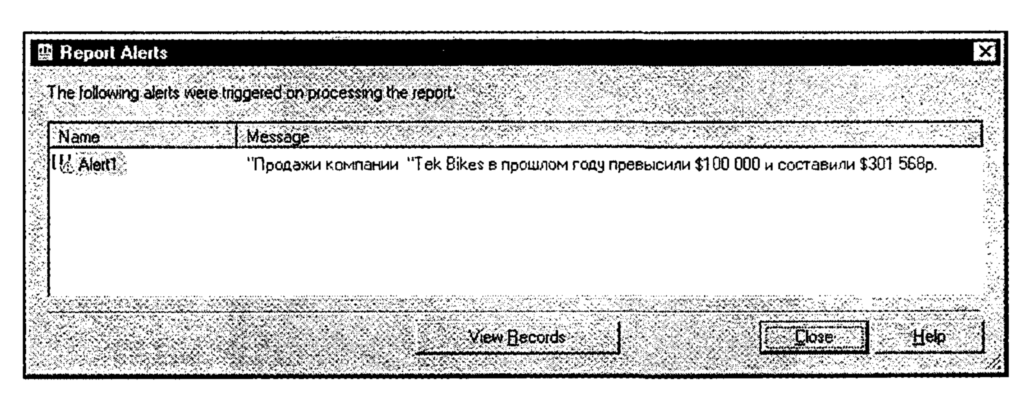


Рис. 5.11. Диалоговое окно **Report Alerts**

Для принудительного вызова специального сообщения следует выполнить команду меню **Report | Alerts | Triggered Alerts**.

Для отмены просмотра сообщения при обновлении отчета надо выключить опцию **Display Alerts on Refresh** на вкладке **Reporting** диалогового окна **Options** (команда меню **File | Options**).

Crystal Reports 9 содержит три функции для работы со специальными сообщениями:

- IsAlertEnabled("AlertName")
- IsAlertTriggered("AlertName")
- AlertMessage("AlertName")

Аргументом этих функций является имя специального сообщения, заключенное в двойные кавычки. Функция AlertMessage("AlertName") возвращает текст специального сообщения. Функция IsAlertTriggered("AlertName") возвращает значение "истина", когда выполняется условие специального сообщения. Функция IsAlertEnabled("AlertName") возвращает значение "истина" (true), если в отчете существует специальное сообщение с именем AlertName.

Функции работы со специальными сообщениями показываются в разделе **Alerts** списка функций окна **Formula Editor**. В этом же разделе содержится список имен специальных сообщений (рис. 5.12).

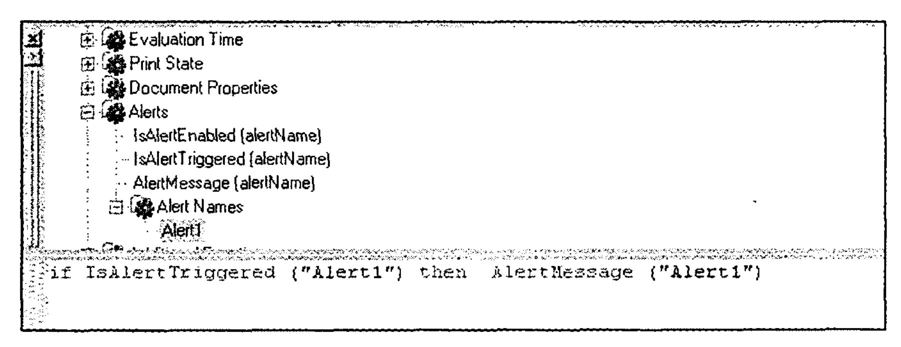


Рис. 5.12. Раздел **Alerts** списка функций окна **Formula Editor**

5.9. Использование параметров в формулах

Параметры могут использоваться в формулах наряду с полями базы данных и специальными полями. Имя параметра начинается с символа ?, например ?Region.

Параметры могут использоваться в формулах, применяющихся для вычислений, выборки и форматирования.

Так, формула, созданная с помощью **Record Selection Formula Editor**

```
{Customer.Region} = {?Region}
```

включит в отчет только те записи, у которых значение поля **Customer.Region** совпадает со значением параметра **?Region**.

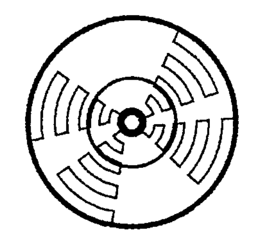
Если параметр принимает одиночное значение, то расположенная в заголовке отчета формула

```
"Отчет по региону" + ?Region
```

вернет это значение, например "Отчет по региону CA". Если параметр принимает множественные значения, то такая формула вернет только первое значение. Для отображения всех значений в заголовке отчета следует использовать формулу:

```
Local NumberVar i;
//функция UBound возвращает размер массива
Local StringVar result := "";
For i :=1 to UBound ({?State})Do
(
result := result+" "+{?State}[i];
);
result
```

Глава 6



Сложное форматирование и специальные типы отчетов

6.1. Форматирование секций

В процессе создания отчетов часто возникает необходимость сложного форматирования секций отчета — иногда надо скрыть или показать ту или иную секцию в зависимости от значений содержащихся в ней полей, выделить секцию цветом, "наложить" одну секцию на другую и т. д.

Для форматирования секции следует расположить курсор на секции в левой части вкладки **Design** окна редактирования отчета и щелкнуть правой кнопкой мыши. В появившемся контекстном меню следует выбрать пункт **Format Section**. Появляется диалоговое окно **Section Expert** (рис. 6.1).

Вкладка **Common** диалогового окна **Section Expert** позволяет выбрать опции форматирования. Выбранные опции могут влиять на отчет непосредственно или же в зависимости от выполнения некоторого условия, см. *разд. 6.2.* Вот список возможностей.

- **Hide** — скрыть секцию с сохранением возможности просмотра детальной информации по выбранной группе (drill down, "высверливание").
- **Suppress** — скрыть секцию без возможности "высверливания".
- **Print at Bottom of Page** — печатать суммирующие значения только в конце страницы.
- **New Page Before** — перейти на новую страницу перед печатью секции.
- **New Page After** — перейти на новую страницу после печати секции.
- **Reset Page Number After** — установить номер страницы в 1 после секции.
- **Keep Together** — не разрывать данные из одной записи на две страницы.
- **Suppress Blank Section** — скрывать пустые секции.
- **Underlay Following Section** — возможность "наложения" секции на следующую секцию.
- **Format with Multiple Column** — возможность многоколоночного форматирования. Эта опция позволяет, в частности, "продлить" отчет на несколько

страниц в ширину, для чего нужно выбрать эту опцию и на появившейся вкладке **Layout** установить ширину **Detail Size** больше ширины страницы. Отчет будет располагаться на нескольких страницах в ширину. Эта опция действует только при форматировании секции **Details**.

- **Read-only** — запрет на форматирование или изменение расположения для всех объектов секции. При включенной опции **Read-only** все другие опции диалогового окна **Section Expert** недоступны.
- **Relative Positions** — устанавливает для объектов секции относительную позицию. Включение этой опции полезно, если в секции размещены объекты, динамически изменяющие свой размер, например матричный отчет (**Cross-Tab**). Если справа от **Cross-Tab** на расстоянии 2 см помещен текстовый объект и включена опция **Relative Positions**, то расстояние в 2 см будет сохраняться как в режиме **Design**, так и в режиме **Preview**. В противном случае **Cross-Tab** может "закрыть" текстовый объект.

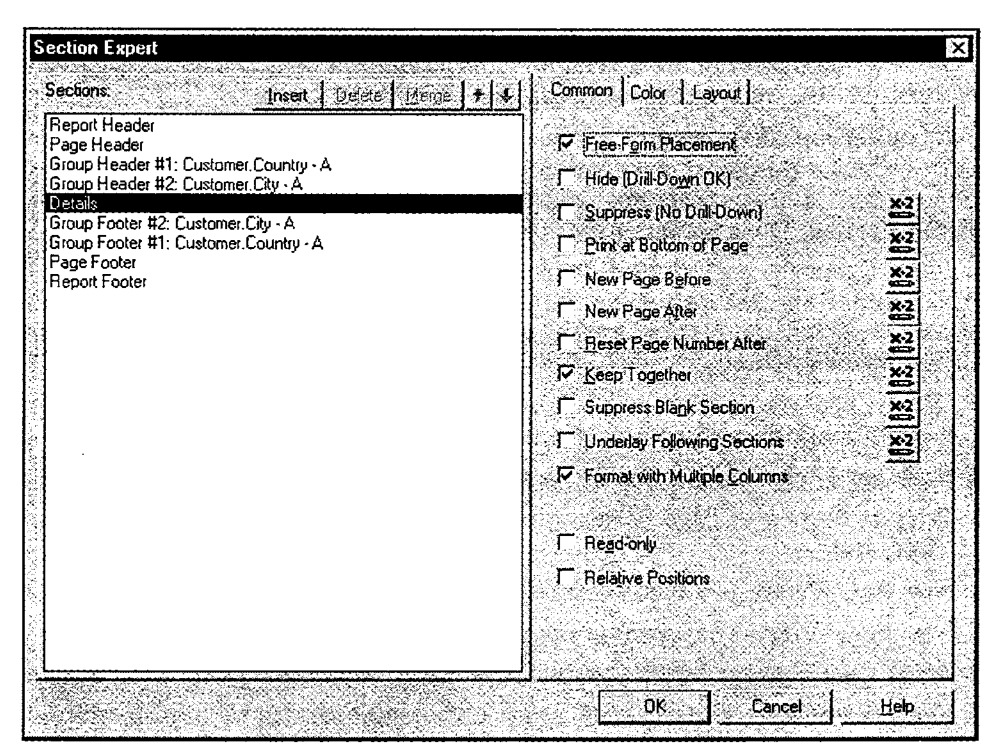


Рис. 6.1. Диалоговое окно **Section Expert**

Рассмотрим возможность "наложения" текущей секции на секцию, следующую за ней. Если в секции **Page Header** расположить рисунок и включить опцию **Underlay Following Section,** то отчет может выглядеть как на рис. 6.2.

Хотя поля **Customer Name** и **Last Year's Sales** расположены в секции **Details**, рисунок располагается не над полями, а справа от них. Секция **Page Header** накладывается на секцию **Details**.

Рис. 6.2. Пример форматирования **Underlay Following Section**

Crystal Reports 9 позволяет создавать множественные секции. Так, например, можно создать две секции **Details** и каждую отформатировать по-своему. Чтобы создать новую секцию, следует в левой части диалога **Section Expert** в списке выбрать секцию, например **Details**, и щелкнуть на кнопке **Insert**. Возникают две секции — **Details a** и **Details b**.

Другой способ создания множественных секций — метод Drag&Drop. Для создания новой секции следует переместить курсор на вертикальную полосу слева от секции, "захватить" ее, при этом возникает курсор в виде двух вертикальных разнонаправленных стрелок, и перетащить ее вниз на некоторое расстояние. Возникает новая секция.

Таким способом можно создать любое количество секций. Каждую секцию можно отформатировать индивидуально, каждая секция может содержать совершенно разные объекты.

Множественные секции позволяют строить очень сложные отчеты прекрасного качества. Рассмотрим несколько примеров использования множественных секций.

- Если в одной секции расположить объекты переменной длины, например матричные отчеты или подотчеты, они могут при печати перекрывать друг друга. Расположение этих объектов в разных секциях обеспечит их правильное расположение при печати (рис. 6.3).
- В отчете может возникнуть необходимость отображения поля в разных форматах. Например, если в секции **Details** есть поля "Страна" и "Дата", дату следует отобразить в таком формате, который принят в данной стране. Можно создать несколько секций **Details**, в каждой из них поместить поле "Страна" и отформатировать его индивидуально и, наконец, в диалоговом окне **Section Expert** установить опцию **Suppress** (скрыть секцию)

5 Зак. 1021

по условию так, чтобы для каждой страны показывалась секция с соответствующим форматом даты.

- Установка различного цвета фона для четных и нечетных строк секции **Details**. Такой способ улучшает восприятие отчетов. Для этого необходимо создать две секции **Details**, установить для них разный цвет фона на вкладке **Color** диалога **Section Expert**, и установить опцию скрыть секцию по условию для четных и нечетных полей.

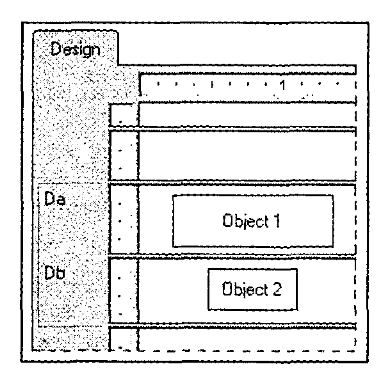


Рис. 6.3. Расположение объектов переменной длины в разных секциях отчета

Упражнение 6.1

Используя в качестве источника данных таблицу **Customer**, создайте отчет с полями **Customer.Last Year's Sales** и **Customer.Customer Name**. Включите в отчет группировку по стране, по полю **Customer.Country**.

Отформатируйте отчет так, чтобы информация о каждой стране располагалась на отдельной странице.



Рис. 6.4. Группа кнопок диалогового окна **Section Expert** для работы с множественными секциями

При создании множественных секций в верхней части диалога **Section Expert** становится доступной группа кнопок, показанная на рис. 6.4. Две правые кнопки этой группы служат для перемещения выделенной в списке секции относительно соседних, например, для изменения порядка множественных секций. Кнопка **Insert**, как было указано ранее, позволяет создать новую секцию. Кнопка **Delete** удаляет секцию со всеми содержащимися в ней полями. Кнопка **Merge** также удаляет секцию, но все содержащиеся в ней поля отчета перемещаются в соседнюю.

6.2. Форматирование по условию и выделение данных с помощью *Highlighting Expert*

Crystal Reports 9 позволяет форматировать объекты отчета по условию, например в зависимости от значения поля отчета. Форматировать по условию можно как секции отчета, так и отдельные объекты. Для форматирования по условию необходимо шелкнуть на кнопке справа от окна выбора в редакторе форматирования, например **Section Expert** для форматирования секции или **Format Editor** для форматирования поля отчета. Появляется окно **Formula Editor**. Если устанавливаемое свойство может принимать значения "да" — "нет", например **Suppress**, то возвращаемое значение формулы должно быть "истина" или "ложь". Если устанавливаемое свойство может принимать несколько значений, например цвет, то формула должна возвращать константу. В этом случае список констант появляется в верхней части списка функций окна **Formula Editor** (рис. 6.5).

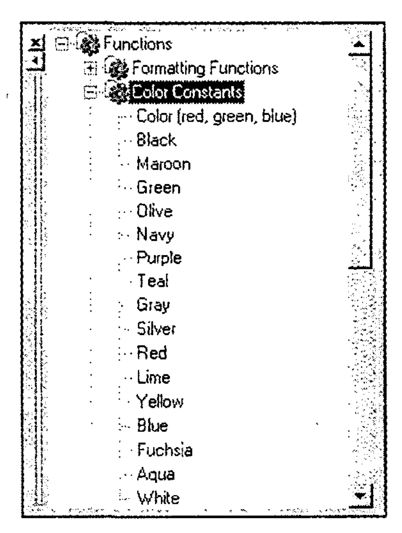


Рис. 6.5. Список констант в окне **Formula Editor**

Упражнение 6.2

Используя в качестве источника данных таблицу **Customer**, создайте отчет и включите в него поля **Customer.Last Year's Sales** и **Customer.Customer Name**. Сделайте красным фон строк, у которых сумма покупок клиентов в прошлом году была больше 30000.

Для этого в режиме дизайна правой кнопкой мыши шелкните в правой части секции **Details** и в диалоговом окне **Section Expert** (рис. 6.1) перейдите

на вкладку **Color**. Щелкните на кнопке справа от поля выбора **Background Color** и в окне **Formula Editor** введите формулу

```
if {Customer.Last Year's Sales}>30000 then crRed else crNoColor
```

Для выделения наиболее значимой информации цветом или стилем шрифта можно использовать специальный инструмент — **Highlighting Expert**. Для выделения объекта отчета следует переместить на него указатель, щелкнуть правой кнопкой мыши и выполнить команду контекстного меню **Highlighting Expert** либо щелкнуть на кнопке вызова **Highlighting Expert** в панели инструментов для анализа (табл. 1.4). Появляется диалоговое окно **Highlighting Expert** (рис. 6.6).

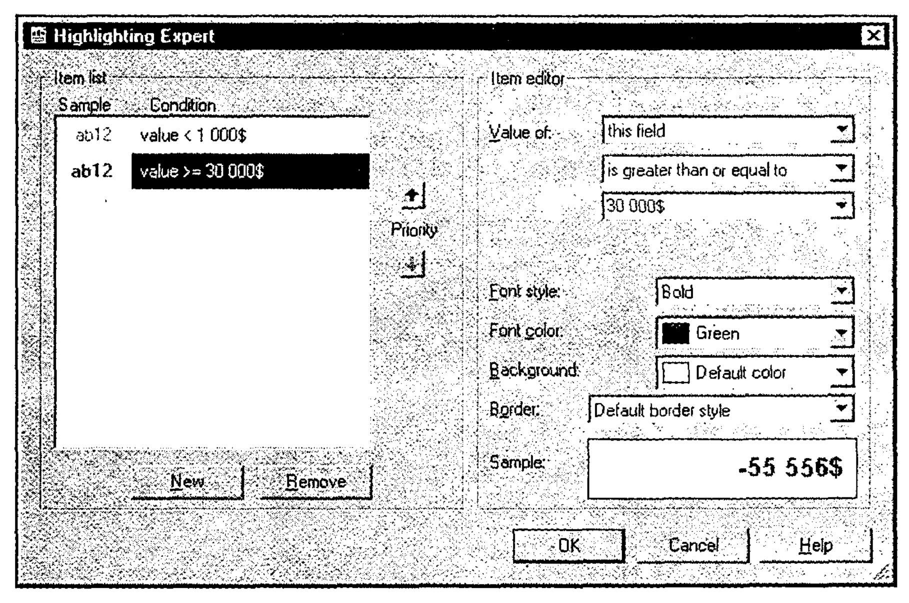


Рис. 6.6. Диалоговое окно **Highlighting Expert**

В левой части диалогового окна **Highlighting Expert** содержится список логических условий форматирования объекта **Item list**. Так, в примере на рис. 6.6 создано два логических условия:

```
Value<1000$
```

и

```
Value>30000$
```

Для каждого логического условия в правой части диалогового окна можно задать предикат и действия, которые будут выполняться, если условие примет значение "истина". Для создания нового условия необходимо щелкнуть на кнопке **New**, для удаления существующего условия — на кнопке **Remove**. В списке может быть задано несколько условий, причем они могут противо-

речить друг другу. В этом случае приоритет будет иметь то условие, которое расположено выше. Для изменения приоритета условий следует воспользоваться кнопками Priority, при этом выделенное условие будет перенесено выше или ниже в списке.

Для задания свойств следует переместить указатель на строку с условием и в группе **Item Editor** выбрать необходимые опции:

Value of — группа из трех раскрывающихся списков, в которых следует выбрать объект отчета, логический оператор и значение. Значение this field в верхнем списке показывает, что условие будет строиться на основе поля, которое форматируется (т. е. было выбрано при вызове диалогового окна **Highlighting Expert**). В примере на рис. 6.2 было выбрано поле **Customer.Last Year's Sales**, оно же будет выделяться цветом. Однако можно, например, форматировать поле **Customer.Customer Name** в зависимости от значения поля **Customer.Last Year's Sales**. В этом случае следует вызвать диалог **Highlighting Expert**, установив указатель на поле **Customer.Customer Name**, а списке **Value** of выбрать **Customer.Last Year's Sales**.

В примере на рис. 6.6 для второго в списке Item list условия выбраны значения:

```
this field
is greater than or equal to
30 000$.
```

Эти значения соответствуют условию **Value>30000$**.

Раскрывающиеся списки **Font style**, **Font color**, **Background** и **Border** позволяют выбрать стиль и цвет шрифта, цвет фона, и стиль рамки объекта отчета, которые будут установлены для объекта, если условие будет выполнено.

В окне **Sample** отображается пример шрифта, установленного с помощью списков **Font style**, **Font color**, **Background** и **Border**.

Упражнение 6.3

Примером форматирования секций по условию может быть отчет, в котором для различных категорий клиентов нужно отображать совершенно разную информацию. Используя в качестве источника данных таблицу **Customer**, создайте отчет и включите в него:

1. Поля **Customer Name**, **Contact Title**, **Contact First Name**, **Contact Last Name**, **Postal Code**, **Address1** и **Address2**, если значение поля **Customer.Last Year's Sales** больше 10000.
2. Поля **Customer Name** и **Last Year's Sales**, если значение поля **Customer.Last Year's Sales** меньше или равно 10000. В качестве источника данных используйте таблицу **Customer**.

Для такого отчета необходимо создать две секции **Details a** и **Details b** *(см. разд. 6.1)*. В секции **Details a** расположите поля **Last Year's Sales**,

Customer Name, **Contact Title**, **Contact First Name**, **Contact Last Name**, **Postal Code**, **Address1** и **Address2**, в секции **Details b** — **Last Year's Sales**, **Customer Name**, **Country** и **City**. Затем следует перейти в диалоговое окно **Section Expert** и для опции **Suppress (No Drill-Down)** создать формулу форматирования.

Для секции **Details a**:

```
{Customer.Last Year's Sales}<=10000
```

Для секции **Details b**:

```
{Customer.Last Year's Sales} >10000
```

Секция **Details a** будет скрыта, если значение поля **Last Year's Sales** меньше или равно 10000, секция **Details a** будет скрыта, если значение поля **Last Year's Sales** больше 10000.

6.3. "Высверливание" данных (Drill Down)

Отчеты могут содержать множество детальной информации и сложное группирование, что может затруднять их восприятие. Например, отчет содержит две колонки — название страны и сумму продаж в каждой стране. При анализе агрегатных данных отчета — сумма продаж в каждой стране — у пользователя может возникнуть необходимость просмотреть продажи в регионах отдельно взятой страны. После просмотра отчета по регионам может потребоваться отчет по продажам в отдельном регионе и т. д. Отчет, включающий только детальную информацию, был бы слишком объемным. Отчет, включающий только суммарную информацию, не позволяет просматривать детали. Для эффективного анализа данных необходимо иметь отчет, сочетающий как суммирующую, так и детальную информацию только там, где это необходимо. Такой отчет должен быть, с одной стороны, компактным, с другой — достаточно подробным. Crystal Reports 9 позволяет скрыть некоторые секции отчета и показывать данные в скрытых секциях на дополнительной вкладке, тем самым, решив проблему детального анализа данных.

Для создания такого отчета следует выполнить группирование и при форматировании секций **Details**, а также, в случае необходимости, секций **Group Header** и **Group Footer**, для внутренних групп в диалоговом окне **Section Expert***включить опцию **Hide (Drill-Down OK)**. После щелчка на кнопке **OK** создается отчет, содержащий только суммирующую информацию. Для просмотра детальной информации в режиме просмотра отчета на вкладке **Preview** можно дважды щелкнуть на заголовке группы. Создается новая вкладка, на которой показывается детальная информация, но не вся, а только соответствующая выбранной группе. В качестве названия вкладки используется имя выбранной группы (рис. 6.7).

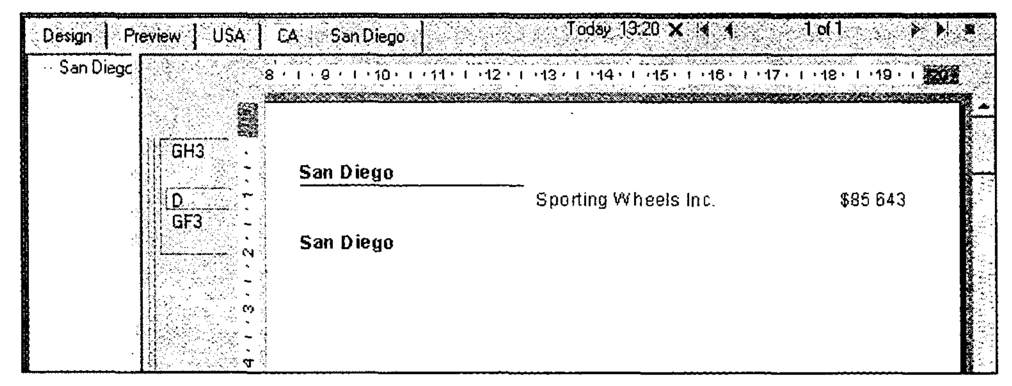


Рис. 6.7. Пример отчета с "высверливанием" (**Drill Down**) данных

Количество вкладок с детальной информацией не ограничено. Вложенность групп, способных создавать вкладки, также не ограничена. Вкладка может соответствовать как заголовку внутренней группы, так и секции **Details**.

В отчетах Crystal Reports 9 возможность "высверливания" данных сохраняется не только для полей отчета, например, заголовка группы, но и для диаграмм или географических карт. Если диаграмма или географическая карта созданы на основе суммирующей информации, то двойной щелчок в сегменте диаграммы или области географической карты приведет к созданию новой вкладки с детальной информацией.

Упражнение 6.4

Используя в качестве источника данных таблицу **Customer**, создайте отчет с полями **Customer.Customer Name** и **Customer.LastYear's Sales**. Сгруппируйте данные по полям **Customer.Country**, **Customer.Region** и **Customer.City**. Скройте, выбрав опцию **Hide (Drill-Down OK)**, секцию **Details** и секции **Group Header** и **Group Footer** для групп **Customer.Region** и **Customer.City**. В режиме просмотра отчета дважды щелкните на названии страны, региона и города.

Для работы с отчетами, содержащими "высверливание" данных Crystal Reports 9 содержит функцию **DrillDownGroupLevel**, которая возвращает целое число — уровень "высверливания". Рассмотрим пример, когда формула, содержащая текст

```
DrillDownGroupLevel
```

размещена в секции **Details**, а отчет содержит три вложенные группы, причем (рис. 6.7) для секции **Details** и двух внутренних групп установлена опция **Hide (Drill-Down OK)**. В этом случае формула вернет значение 3.

Использование функции **DrillDownGroupLevel** может быть полезно для проверки, является ли текущее окно результатом "высверливания". Тогда формула

```
DrillDownGroupLevel>0
```

вернет значение "истина", если применялось "высверливание".

6.4. Этикетки (Mail Label)

Помимо стандартного отчета, диалоговое окно **Report Expert** *(разд. 2.1)* позволяет создать три других специальных типа отчетов: Cross-Tab, OLAP и Mail Label. Для создания специального отчета в диалоговом окне **Crystal Reports Gallery** (рис. 2.1) следует выбрать необходимый тип отчета. Содержание появляющегося диалогового окна **Report Expert** зависит от типа выбранного отчета.

Отчет печати этикеток (Mail Label) позволяет создавать этикетки или печатать адреса на конвертах на основе информации, хранящейся в базе данных. После выбора типа отчета **Mail Label** в диалоговом окне **Crystal Reports Gallery** открывается мастер отчетов **Mailing Labels Report Creation Wizard**.

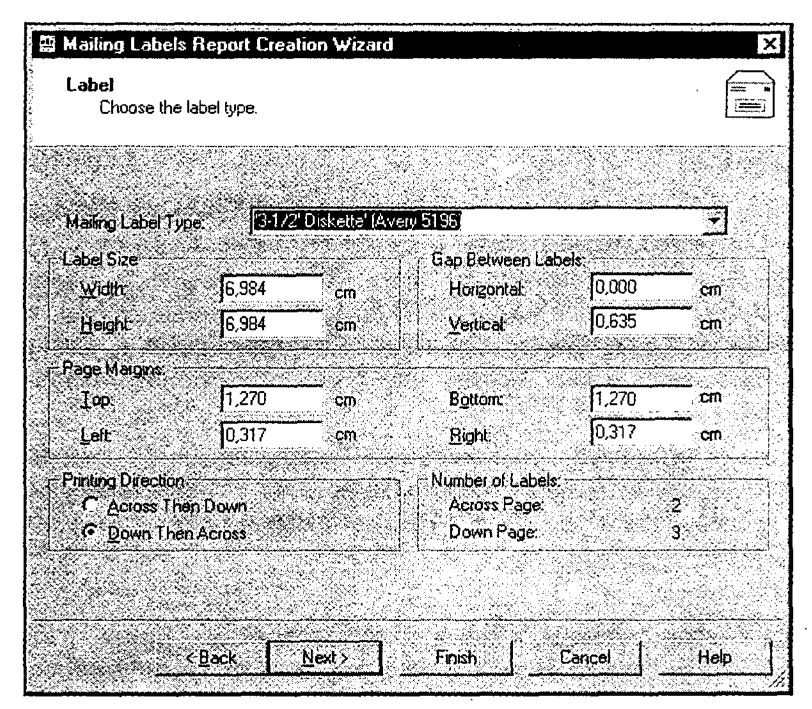


Рис. 6.8. Окно **Label** мастера **Mailing Labels Report Creation Wizard**

Окна мастера — **Data**, **Link**, **Fields** и **Record Selection** не отличаются от соответствующих окон мастера **Standard Report Creation Wizard** (см. рис. 2.2–2.4 и рис. 2.9). В них следует выбрать источники данных, которые будут использованы для создания отчета, связать таблицы и задать условия выборки данных.

В окне **Label** мастера **Mailing Labels Report Creation Wizard** (рис. 6.8) можно выбрать тип этикетки или создать этикетку собственного формата. Раскрывающийся список **Mailing Label Type** содержит набор форматов для конвертов, товарных меток, форматы этикеток для видео- и аудиокассет, дискет и т. д.

Поля раздела **Number of Labels** показывают, сколько этикеток будет расположено вдоль (**Down Page**) и поперек (**Across Page**) страницы. Эта информация вычисляется автоматически в зависимости от размера страницы, рамок и т. д.

Группа **Page Margin** позволяет регулировать отступы от краев страницы.

Опция **Label Size** позволяет регулировать размер создаваемой этикетки — высоту и ширину.

Группа **Gap Beetwin Labels** позволяет задать интервалы между соседними этикетками по вертикали (**Vertical**) и горизонтали (**Horizontal**).

6.5. Матричный отчет (Cross-Tab)

Очень часто данные хранятся в таблицах базы данных в неудобной для создания отчета форме. Рассмотрим, например, таблицу "Квартальные платежи", содержащую три поля — "Плательщик", "Квартал" и "Сумма платежа" (табл. 6.1).

Таблица 6.1. *Пример хранения данных в таблице* ***Квартальные платежи***

Плательщик	Квартал	Сумма платежа
Иванов	1	200
Петров	2	300
Петров	3	100
Сидоров	2	200
Иванов	2	300
Иванов	3	200

Предположим, что перед нами стоит задача создать отчет следующего вида (табл. 6.2).

Таблица 6.2. Пример матричного отчета

Плательщик	Сумма платежа за 1 квартал	Сумма платежа за 2 квартал	Сумма платежа за 3 квартал	Сумма платежа за 4 квартал
Иванов	200	300	200	
Петров		300	100	
Сидоров		200		

В отчете (табл. 6.2) необходимо представить данные в компактном матричном формате, как бы "развернуть" данные в матрицу. Такое представление данных характерно для электронных таблиц. Отчет, по структуре совпадающий с этой таблицей, можно создать, используя формулы Crystal Reports 9, но создание отчета с применением формул трудоемко и может потребовать много времени. Еще более сложная задача возникает, если количество колонок произвольное, а не по количеству кварталов, как в примере.

Для создания матричных отчетов Crystal Reports 9 имеет специальный инструмент — объект **Cross-Tab**.

Cross-Tab — матричный объект, который выглядит как электронная таблица. Этот объект содержит минимум три поля базы данных:

1. Поле строки.
2. Поле столбца.
3. Суммарное поле.

Для создания матричного отчета можно воспользоваться мастером **Cross-Tab Report Creation Wizard**, который вызывается из **Crystal Reports Gallery** (рис. 2.1). Другой вариант — вставить в обычный отчет, который уже содержит секцию **Details**, а также одну или более групп, объект **Cross-Tab**, выполнив команду меню **Insert | Cross-Tab** либо нажав кнопку на панели инструментов для вставки объектов в отчет (табл. 1.3). Вставленный объект становится частью отчета. Матричные отчеты могут быть помещены либо в заголовок (**Page Header**) или подвал страницы (**Page Footer**), но не в секцию **Details**. Форма курсора будет указывать на невозможность такого действия при попытке поместить матричный отчет в эту секцию.

Окна мастера **Cross-Tab Report Creation Wizard — Data**, **Link** и **Record Selection** идентичны соответствующим окнам мастера **Standard Report Creation Wizard** (см. рис. 2.2, 2.3 и 2.9). В них следует выбрать источники данных, которые будут использованы для создания отчета, связать таблицы и задать условия выборки данных.

Окно **Cross-Tab** (рис. 6.9) содержит окна для формирования полей, которые составят строки, столбцы и суммирующие поля матричного отчета. Для вне-

сения поля базы данных в матричный отчет, в списке окна **Available Fields** следует выбрать поле и затем щелкнуть на кнопке > слева от окна **Rows** для использования поля в качестве строки матричного отчета, **Columns** — колонки, **Summarized Fields** — суммирующего поля. Можно также добавить поля в матричный отчет, просто перетащив их методом Drag&Drop из окна **Available Fields** в окна **Rows**, **Columns** или **Summarized Fields**.

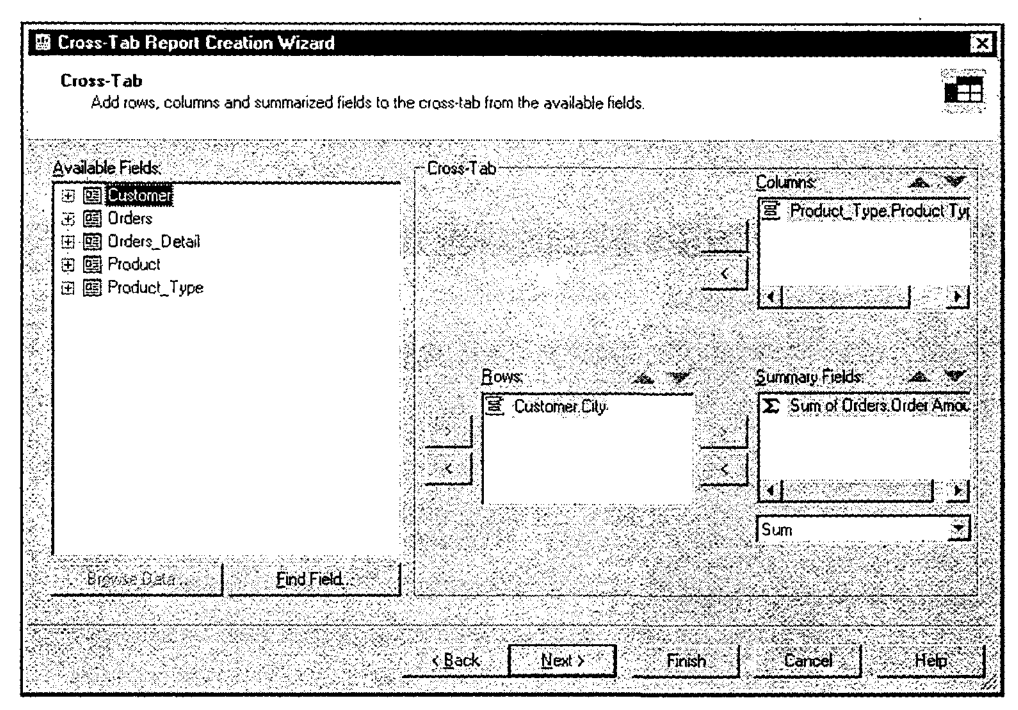


Рис. 6.9. Окно **Cross-Tab** мастера **Cross-Tab Report Creation Wizard**

В матричный отчет в качестве столбцов или строк могут быть включены несколько полей баз данных. В этом случае строки и столбцы выстраиваются иерархически, например, если в качестве столбца выбраны **Country**, **Region** и **City**. При этом создаются групповые связи между полями, в одной стране — несколько регионов, в одном регионе — несколько городов. Конечно, поля должны располагаться в правильном порядке: верхнее — **Country**, затем **Region** и нижнее — **City**. Если порядок неправилен, поле можно переместить в окнах **Rows** и **Columns** методом Drag&Drop. Иногда требуется показать несколько независимых колонок или строк в матричном отчете, например отобразить стоимость, плановую стоимость и дисперсию. К сожалению, отображение независимых колонок на одном уровне невозможно, поскольку множественные поля **Rows** или **Columns** в матричном отчете Crystal Reports 9 всегда показываются в групповой иерархии.

Для каждой группы будет вычисляться собственная сумма. Суммирующих полей также может быть несколько, например **Quantity Sold** (Количество продаж) или **Order Amount** (Сумма заказов). Матричный отчет просто включит два числа в каждую ячейку. Если в качестве суммирующего поля выбирается числовое поле, Crystal Reports 9 вычисляет сумму поля для каждой ячейки с помощью функции **Sum**; если поле с другим типом данных — будет показано количество одинаковых значений поля для каждой комбинации строка/столбец с помощью функции **Count**. Если нужно выбрать агрегатную функцию, отличную от функции по умолчанию, например, среднюю величину продаж, то следует раскрыть список снизу от окна **Summarized Fields**.

Окно **Chart** (рис. 6.10) предназначено для создания диаграммы на основе матричного отчета.

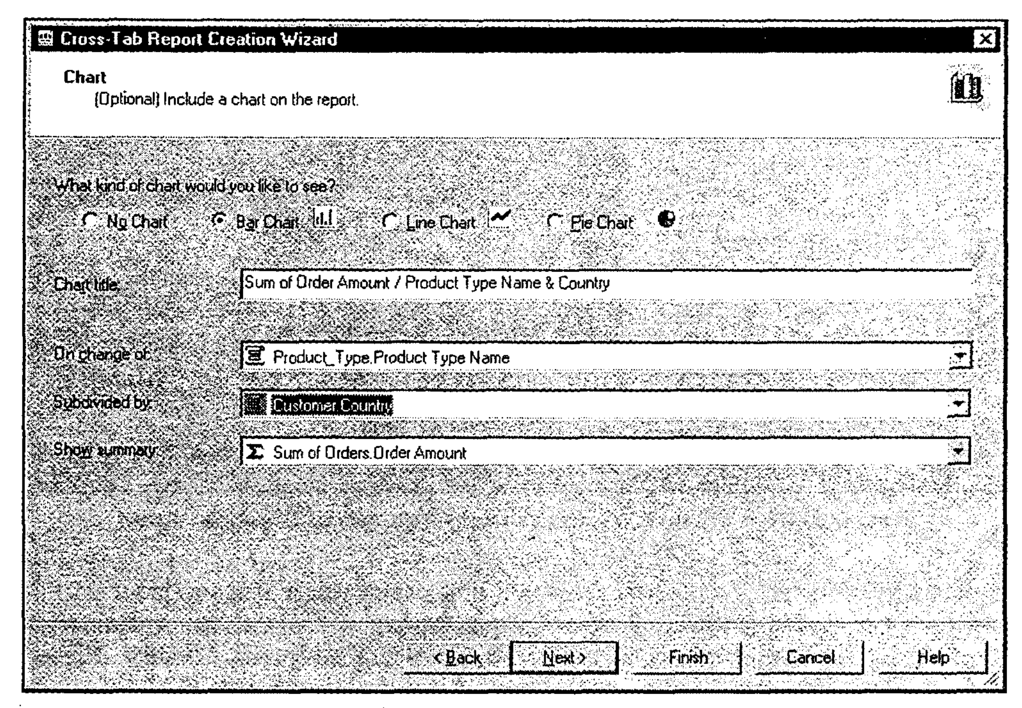


Рис. 6.10. Окно **Chart** мастера **Cross-Tab Report Creation Wizard**

В верхней части окна следует выбрать один из трех типов диаграммы: **Bar**, **Line** или **Pie**. Важно выбрать наиболее подходящий для отображения матричного отчета тип диаграммы. Так, диаграмма типа **Bar** может иметь три размерности — строка, столбец и суммирующее поле. В раскрывающихся списках **On change of**, **Subdivided by** и **Show summary** можно указать поля отчета, которые будут использоваться для размерностей диаграммы. В поле **Chart title** следует ввести заголовок диаграммы.

В окне **Grid Style** (рис. 6.11) можно выбрать один из семнадцати предопределенных стилей таблицы матричного отчета.

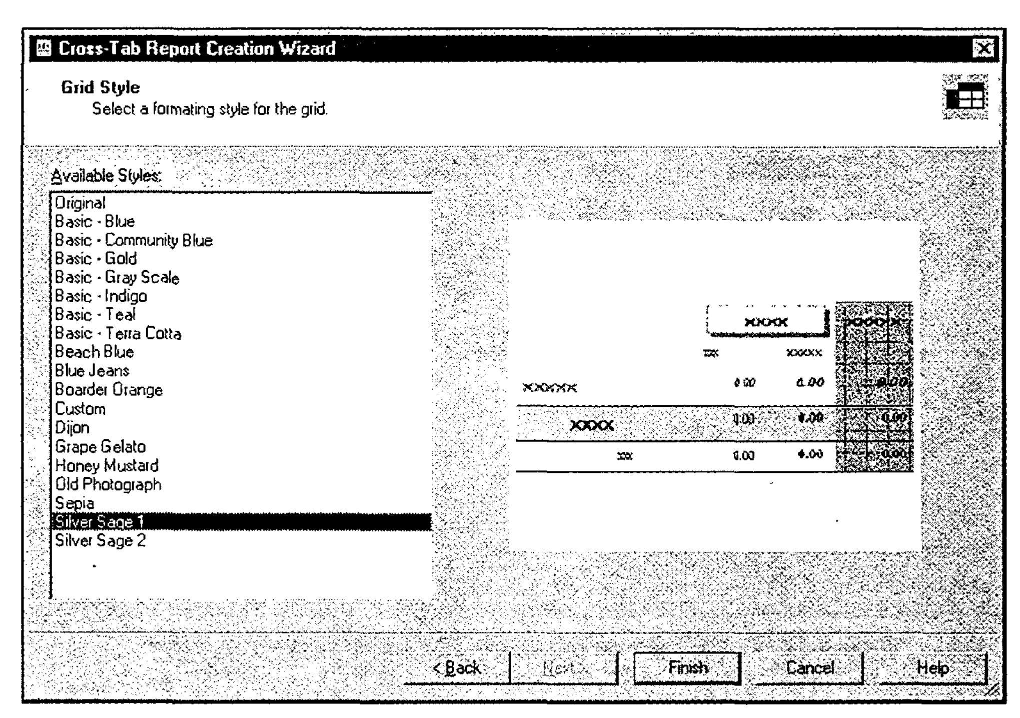


Рис. 6.11. Окно **Grid Style** мастера **Cross-Tab Report Creation Wizard**

После щелчка на кнопке **Finish** создается отчет, содержащий объект **Cross-Tab**.

Для внесения объекта **Cross-Tab** в существующий отчет следует выполнить команду меню **Insert | Cross-Tab** или щелкнуть на кнопке вставки матричного отчета в панели инструментов для вставки объектов в отчет (табл. 1.3), после чего переместить указатель на секцию отчета, в которую нужно вставить **Cross-Tab**, и щелкнуть кнопкой мыши.

Для редактирования визуального отображения существующего объекта **Cross-Tab** следует выделить его целиком, а не только одно из полей, входящих в его состав. Для этого нужно щелкнуть мышью либо в небольшой незаполненной области в левом верхнем углу **Cross-Tab** (над первой строкой и слева от первого столбца), либо на линиях сетки между ячейками. После этого необходимо выполнить команду контекстного меню **Format Cross-Tab** или в команду главного меню **Format | Format Cross-Tab**. Появляется диалоговое окно **Format Editor**, которое идентично диалоговому окну для редактирования свойств полей отчета (рис. 2.12).

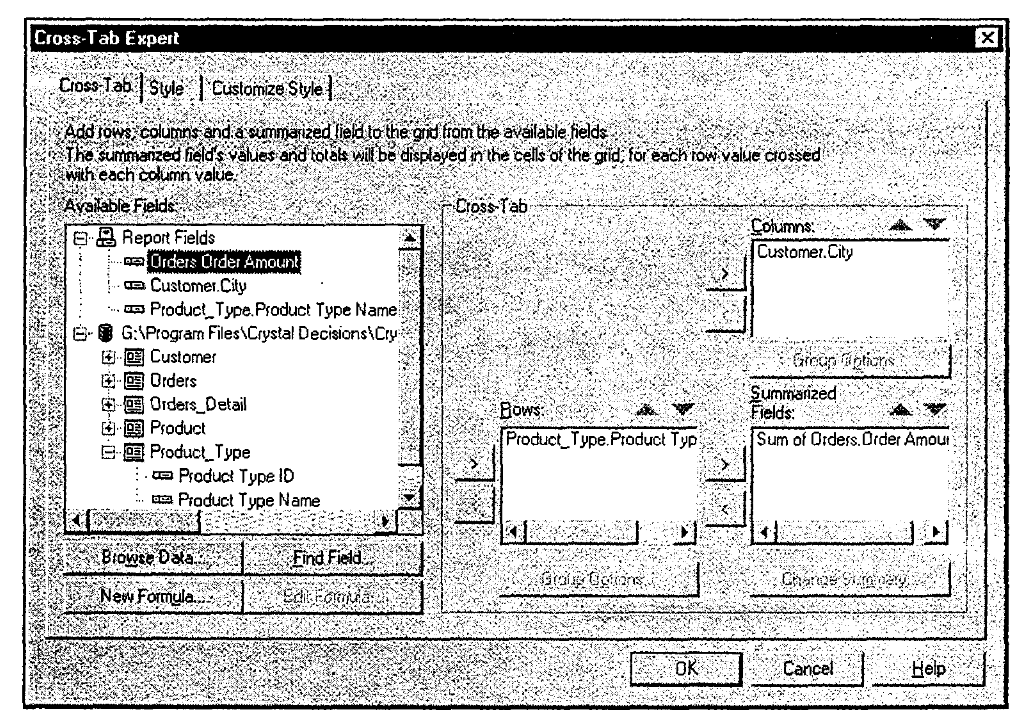


Рис. 6.12. Вкладка **Cross-Tab** диалогового окна **Cross-Tab Expert**

Для изменения свойств созданного объекта **Cross-Tab** следует щелкнуть по нему правой кнопкой мыши и в контекстном меню выбрать пункт **Cross-Tab Expert**. Возникает диалоговое окно **Cross-Tab Expert** с вкладками **Cross-Tab**, **Style**, **Customize Style**.

Появляется диалоговое окно **Cross-Tab Expert**, имеющее три вкладки — **Cross-Tab**, **Style** и **Customaze Style**. Вкладка **Cross-Tab** (рис. 6.12) так же, как окно **Cross-Tab** мастера **Cross-Tab Report Creation Wizard** служит для выбора полей базы данных, на основе которых создается **Cross-Tab**. Но в отличие от мастера **Cross-Tab Report Creation Wizard** здесь можно использовать не только поля базы данных, но и формулы. Например, в базе данных содержится поле с датой заказа **Orders.Order Date**. Необходимо создать матричный отчет с суммированием по кварталам. Для этого нам нужно создать формулу (**Quarter**), которая определит квартал заказа, кнопкой **New Formula** (рис. 6.12). Текст этой формулы следующий:

```
If month ({Orders.Order Date }) in 1 to 3 then "1st Quarter"
else if month ({Orders.Order Date }) in 4 to 6 then "2nd Quarter"
else if month ({Orders.Order Date }) in 7 to 9 then "3rd Quarter"
else "2rd Quarter"
```

Во вкладке **Cross-Tab** можно изменить агрегатную функцию суммарного поля. Для этого необходимо выбрать суммарное поле в списке суммарных полей **Summarized Fields** в правой нижней части вкладки **Cross-Tab** и щелкнуть на кнопке **Change Summary**. Появляется диалоговое окно **Change Summary** (рис. 6.13), в котором можно выбрать другую агрегатную функцию. Если выбрать опцию **Show as a percentage of**, то в матричном отчете будет отображаться не абсолютное значение суммирующего поля, а его доля в процентах от величины, выбранной в раскрывающемся списке в нижней части диалогового окна **Change Summary**. Заметим, что в матричных отчетах суммарные поля (**Summarized Field**) могут быть только числовыми и не могут иметь тип "текст" или "дата".

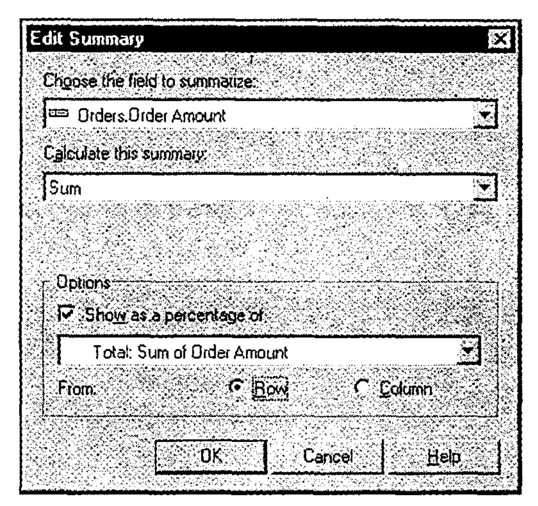


Рис. 6.13. Диалоговое окно **Change Summary**

При формировании матричного отчета Crystal Reports 9 группирует данные по полям, указанным в качестве строк и столбцов. Для изменения порядка группировки следует на вкладке **Cross-Tab** диалогового окна **Cross-Tab Expert** выделить поле строки или столбца, которое необходимо изменить, и щелкнуть на кнопке **Group Options**. Появляется диалоговое окно **Cross-Tab Group Options** (рис. 6.14).

Диалоговое окно **Cross-Tab Group Options** аналогично окну **Insert Group** (рис. 3.6), используемому для задания порядка группировки в стандартном отчете. Для поля, отличного от поля даты, доступны три опции:

- **Ascending Order** — отображение строки или столбца матричного отчета по алфавиту в порядке возрастания;
- **Descending Order** — отображение строки или столбца матричного отчета по алфавиту в порядке убывания;

- **Specified Order** — возможность создавать строки или столбцы на основании содержимого полей базы данных, отобранных для строки или столбца.

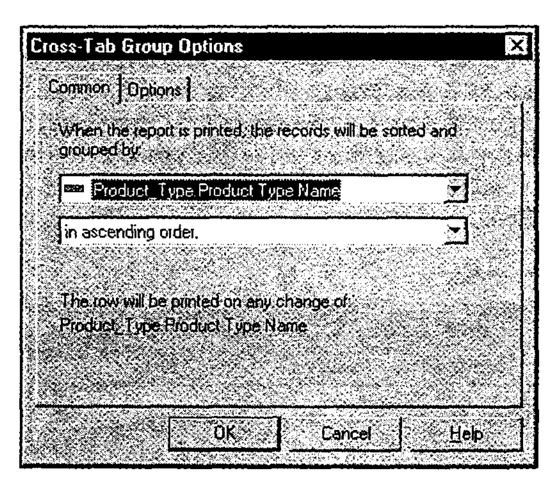


Рис. 6.14. Диалоговое окно **Cross-Tab Group Options**

Рис. 6.15. Вкладка **Customize Style** диалогового окна **Cross-Tab Expert**

Если поле строки или столбца является полем даты, времени или даты/времени, диалоговое окно **Cross-Tab Group Options** предлагает дополнительные опции. Так, например, в одну группу могут быть объединены записи по значению поля типа дата за один день, одну неделю, месяц, квартал, год и т. д. (**The column (row) will be printed**). Раскрывающийся список **The value printed for the column (row) will be** предоставляет две возможности выбора: начальная дата периода и конечная дата периода.

Вкладка **Style** идентична окну **Grid Style** мастера **Cross-Tab Report Creation Wizard** (рис. 6.11).

Для дополнительного форматирования объекта **Cross-Tab** необходимо использовать вкладку **Customize Style** (рис. 6.15).

В верхней части вкладки **Customize Style** расположены окна **Rows** и **Columns**. Одновременно можно выбрать только один объект (строку или столбец матричного отчета) в одном из двух окон. Именно для этого объекта можно установить свойства с помощью элементов управления, расположенных в нижней части вкладки. В табл. 6.3 приведены опции форматирования строк и столбцов матричного отчета.

Таблица 6.3. *Опции форматирования строк и колонок матричного отчета*

Опция	Описание опции
Suppress Subtotal	Скрывает промежуточную сумму по строкам или столбцам для выбранного поля, если создана иерархия по строкам или столбцам, т. е. в качестве строк или столбцов выбрано более одного поля базы данных
Suppress Label	Опция становится доступной, если выбрана опция **Suppress Subtotal**. Скрывает поле верхнего уровня в иерархии
Alias for Formulas	Используется для ссылки на целую строку или столбец при выполнении условного форматирования в матричном отчете
Background Color	Устанавливает цвет фона для строки или столбца
Show Cell Margins	Окружает ячейки свободным пространством со всех сторон. Отключение этой опции позволяет размещать ячейки более компактно
Indent Row Labels	Выбор этой опции приводит к отступу подписи (Label) выбранной строки от левого края матричного отчета. Величина отступа строки задается в поле снизу от окна выбора **Indent Row Labels**
Format Grid Lines	Вызывает диалоговое окно **Format Grid Lines**, которое предназначено для задания формата линий сетки в матричном отчете
Keep Columns Together	Защищает столбцы от разбиения, когда распечатываемая таблица превосходит по ширине страницу и таблица должна быть распечатана на двух или более страницах

Таблица 6.3 (окончание)

Опция	Описание опции
Repeat Row Labels	При выборе **Keep Columns Together** повторяет метки строки, если ширина матричных отчетов превышает ширину страницы
Row Totals on Left	Отображает итог по строкам слева, а не справа строк
Column Totals on Top	Отображает итог по столбцам вверху, а не внизу от текущей колонки
Suppress Empty Rows	Скрывает строки, не содержащие данных
Suppress Empty Columns	Скрывает столбцы, не содержащие данных
Suppress Row Grand Total	Скрывает общий итог строки
Suppress Column Grand Total	Скрывает общий итог столбца

Объект **Cross-Tab** можно также "транспонировать", поменяв местами строки и столбцы. Для этого нужно выполнить команду главного меню **Format | Pivot Cross-Tab** или щелкнуть правой кнопкой мыши на **Cross-Tab** и выполнить команду контекстного меню **Pivot Cross-Tab**.

Матричный отчет создается в несколько этапов. На каждом этапе производятся вычисления лишь по одной комбинации строки и столбца, поэтому формируется матричный отчет сравнительно медленно. Если поместить объект **Cross-Tab** в заголовок или подвал отчета (секции **Report Footer** и **Report Header**), то он появится в отчете только в одном экземпляре. Этот **Cross-Tab** будет обобщать все данные отчета. Если поместить объект **Cross-Tab** в заголовок или подвал группы (секции **Group Footer** и **Group Header**), то в результате объектов будет столько, сколько существует групп, и в каждый **Cross-Tab** будут включены данные только этой группы.

После внесения в секцию отчета, на вкладке **Design** объект **Cross-Tab** выглядит как набор дочерних полей (рис. 6.16):

- суммирующее поле;
- заголовок столбца, отображаются поля базы данных;
- заголовок суммы по колонкам — текстовый объект;
- заголовок строки, отображаются поля базы данных;
- заголовок суммы по строкам — текстовый объект;
- сумма по столбцам;
- сумма по строкам;
- итоговая сумма.

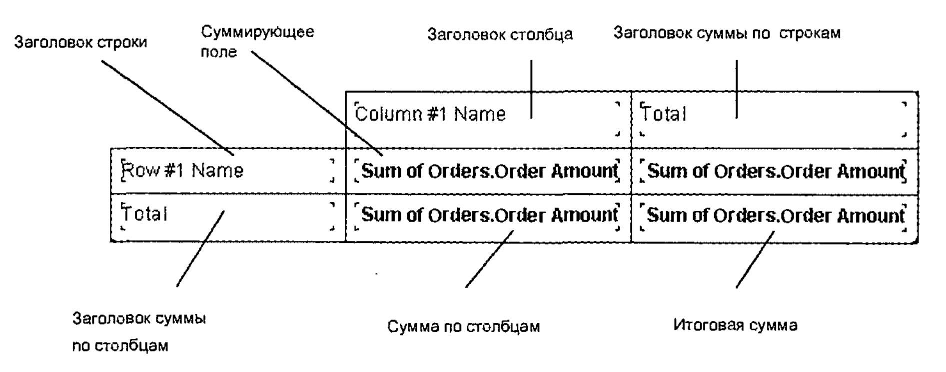


Рис. 6.16. Представление объекта **Cross-Tab** в режиме **Design**

Каждый дочерний объект **Cross-Tab** может быть отформатирован индивидуально, в том числе по условию, как обычное поле отчета в редакторе **Format Editor**. Размер дочернего объекта можно изменить так же, как размер обычного поля. Можно выделить несколько объектов в матричном отчете, а затем отформатировать их все одновременно. Для условного форматирования используется диалоговое окно **Highlighting Expert**, вызывается щелчком правой кнопки на объекте и выполнением команды контекстного меню **Highlighting Expert**. Также для условного форматирования используются формулы, они содаются в диалоговом окне **Formula Editor**, которое вызывается кнопкой условного форматирования в диалоговом окне **Format Editor**. При создании формул условного форматирования матричного отчета нельзя использовать поля базы данных или поля промежуточных сумм матричного отчета. При условном форматировании матричных отчетов необходимо использовать встроенную функцию **Current Field Value**, которая возвращает текущее значение форматируемого поля. Например, приведенная ниже формула позволяет окрасить поле матричного отчета в красный цвет, если значение поля больше 20000:

```
if CurrentFieldValue > 20000 then Red else NoColor.
```

Для условного форматирования на основе суммы по строке или столбцу, а не только на основе значения поля, следует использовать функцию **GridRowColumnValue**. С помощью функции **GridRowColumnValue** можно определить строку или столбец, в котором находится поле, например

```
if GridRowColumnValue("Customer.Region") = "CA" then Red else NoColor
```

В качестве аргумента функции **GridRowColumnValue** допускается использование не имени поля, а его псевдонима. Псевдоним задается в поле **Alias for Formulas** вкладки **Customize Style** диалога **Format Cross-Tab**.

На основе данных матричного отчета можно создать диаграмму. Для этого нужно выделить объект **Cross-Tab** и выполнить команду меню **Insert | Chart**. Диаграмма, построенная на основе **Cross-Tab**, должна иметь три размерности:

строка, столбец и суммирующее поле. На вкладке **Type** диалога **Chart Expert** необходимо выбрать подходящий тип диаграммы — **3D Riser** или **3D Surface**. На вкладке **Data** содержатся списки выбора **On change of**, **Sybdivided by** и **Show**, в которых следует указать поля матричного отчета, использующиеся в качестве размерности диаграммы. Как правило, в списках **On change of** и **Sybdivided by** выбирается строка и столбец, а в качестве **Show** — суммирующее поле.

Упражнение 6.5

В файле xtreme.mdb имеются таблицы **Customer** и **Product Type**. В таблице **Customer** содержится информация о городах проживания клиентов, в таблице **Product Type** — наименование типов продаваемых продуктов. Необходимо создать отчет, показывающий объем продаж типов продуктов в городах Канады. Таблицы **Customer** и **Product Type** связаны между собой через три промежуточные таблицы — **Orders**, **Orders Detail** и **Product**. Сумма продаж определяется как сумма заказов в поле **Orders.Order Amount**, сделанных по каждому типу продуктов в каждом регионе.

Для создания отчета следует выполнить следующие шаги:

1. При создании нового отчета выбрать тип отчета **Cross-Tab**.
2. Выбрать источник xtreme.mdb.
3. Ввести таблицы **Customer**, **Orders**, **Orders Detail**, **Product**, **Product Type** и связать их на вкладке **Links**, по умолчанию связи автоматически устанавливаются корректно.
4. В окне **Cross-Tab** в качестве строки **Rows** выбрать поле **Customer.Customer City**, в качестве столбца **Columns** — **Product Type.Product Type Name** и в качестве суммирующего поля **Summarized Fields** — **Orders.Order Amount**.
5. В окне **Chart** выбрать тип диаграммы **Bar Chart** и указать размерности диаграммы: **On change of** — **Customer City**, **Sybdivided by** — **Product Type Name**, **Show summary** — **Sum of Orders.Order Amount**.
6. В окне **Record Selections** задать правила отбора данных:

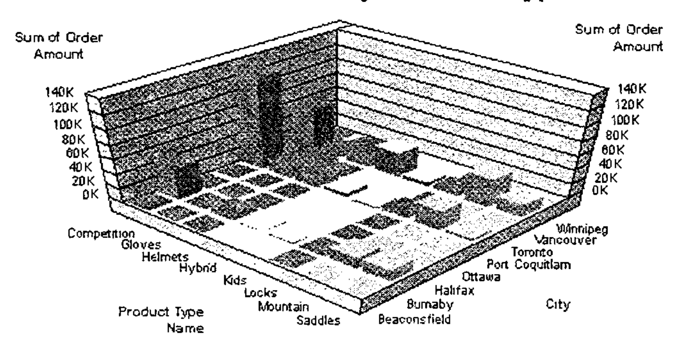
   ```
   Customer.Country is equal to "Canada"
   ```

   и

   ```
   Product Type Name is not equal to ""
   ```

7. После щелчка на кнопке **Finish** создается матричный отчет. Следует отметить, что тип диаграммы **Bar Chart** нельзя назвать удачным для отображения данных матричного отчета. Для изменения типа диаграммы надо щелкнуть на ней правой кнопкой, выполнить команду контекстного меню **Chart Expert** и на вкладке **Type** диалогового окна **Chart Expert** выбрать тип **3D Riser**.

Результат выполнения упражнения показан на рис. 6.17.



	Competiti	Gloves	Helmets	Hybrid	Kids	Locks	Mountain	Saddles
Beaconsfield	1 800$	0$	0$	0$	0$	0$	0$	0$
Burnaby	44 471$	1 002$	9 109$	1 458$	1 344$	149$	16 028$	16 884$
Halifax	0$	0$	0$	0$	0$	0$	960$	0$
Ottawa	0$	0$	0$	0$	274$	0$	0$	0$
Port Coquitlam	126 963$	17 621$	40 477$	6 932$	1 224$	7 349$	27 944$	3 607$
Toronto	8 820$	0$	0$	0$	0$	0$	0$	0$
Vancouver	58 756$	9 236$	25 142$	35 336$	6 410$	5 706$	39 461$	18 250$
Winnipeg	0$	0$	0$	0$	0$	0$	864$	0$
Total	240 810 р.	27 860 р.	74 729 р.	43 726 р.	9 253 р.	13 204 р.	85 256 р.	38 741 р.

Рис. 6.17. Пример матричного отчета, содержащего диаграмму

6.6. Отчет, содержащий подотчеты (Subreport)

Подотчет (subreport) — это отчет, входящий в состав другого отчета. Подотчет является полноценным отчетом и может быть связан или не связан с основным отчетом. В качестве подотчета можно использовать существующий отчет или создать новый. Подотчет может быть включен в любую секцию отчета как объект **Subreport**. Хотя основной отчет может содержать любое количество подотчетов, подотчет не может содержать другой подотчет.

Подотчеты позволяют представить данные с разных точек зрения или связать данные из разных источников. Например, основной отчет может содержать группу по регионам, а подотчет — группу Top N и диаграмму, иллюстрирующую распределение продаж в нескольких крупнейших регионах.

Другой пример использования подотчетов — связывание данных, хранящихся в разных базах данных. Например, в результате перехода на новую информационную систему часть данных предприятия хранится в СУБД Oracle, часть — в DBF-файлах. Основной отчет может использовать в качестве источника Oracle, подотчет, размещенный в секции **Details**, — файл типа DBF. Данные основного отчета и подотчета могут быть связаны по значению поля базы данных, по формуле или по параметру. Например, если табельный номер сотрудника, хранящийся в Oracle, — целое число, а в файле DBF — строка, то отчет и подотчет могут быть связаны с помощью формул. Кроме отчета с подотчетами, Crystal Reports 9 не имеет средств создания отчета из разнородных источников данных.

Для включения подотчета в существующий отчет следует выполнить команду меню **Insert | Subreport** или щелкнуть на кнопке вставки подотчета в панели инструментов для вставки объектов в отчет (табл. 1.3), после чего щелкнуть кнопкой мыши на любой секции отчета.

Возникает диалоговое окно **Insert Subreport**, которое содержит две вкладки — **Subreport** (рис. 6.18) и **Link**.

Рис. 6.18. Вкладка **Subreport** диалогового окна **Insert Subreport**

На вкладке **Subreport** можно указать существующий отчет, который будет использоваться в качестве подотчета, для этого используется радиокнопка **Choose a report**, поле **Report File Name** и кнопка **Browse**. Можно создать новый отчет, используя радиокнопку **Create a subreport**, поле **Report File Name** и кнопку **Report Wizard**. Для создания нового отчета необходимо ввести

в поле его имя и щелкнуть на кнопке **Report Wizard**, которая вызывает мастера отчетов **Standard Report Creation Wizard** *(разд. 2.1)*. Если основной отчет и подотчет связаны, то выполнение отчета может занять много времени, поскольку связывание данных производит Crystal Reports 9, а не сервер базы данных, и индексы при этом не употребляются. Для увеличения производительности при отображении отчета с подотчетом на экране используется опция **On-demand subreport**. Если эта опция включена, то в режиме предварительного просмотра отображается только имя подотчета, а не его содержимое, то есть подотчет не выполняется (рис. 6.19).

09.05.03

Customer Name | Last Year's Sales

PA

Tek Bikes 301 568$

Orders

Clean Air Transportation Co. 23 789$

Orders

Backpedal Cycle Shop 25 162$

Orders

Insane Cycle 8 000$

Orders

Rocky Roadsters 28 682$

Orders

387 201 р.

Рис. 6.19. Пример отчета с включенной опцией **On-demand subreport**

Для просмотра содержимого подотчета необходимо дважды щелкнуть на его имени. При этом Crystal Reports 9 создает новую вкладку, на которой отображается содержимое подотчета, т. е. отчет с подотчетом выполняется так же, как отчет с "высверливанием" данных **Drill Down**.

Вкладка **Links** (рис. 6.20) содержит два списка — список полей отчета и базы данных **Available Fields** и список полей, по которым следует произвести связывание отчета и подотчета **Field(s) to link to**. После выбора поля отчета в нижнем левом списке выбора **Select data in subreport based on field** появляется список полей подотчета того же типа, что и выбранное поле основного отчета. Для связывания отчета и подотчета необходимо в список **Field(s) to link to** ввести имя поля основного отчета, а в нижнем левом списке выбрать соответствующее поле подотчета. Если отчет и подотчет связаны, Crystal Reports 9 автоматически создает параметр и условие выборки по этому параметру, например:

```
{Orders.Customer ID} = {?Pm-Customer.Customer ID}
```

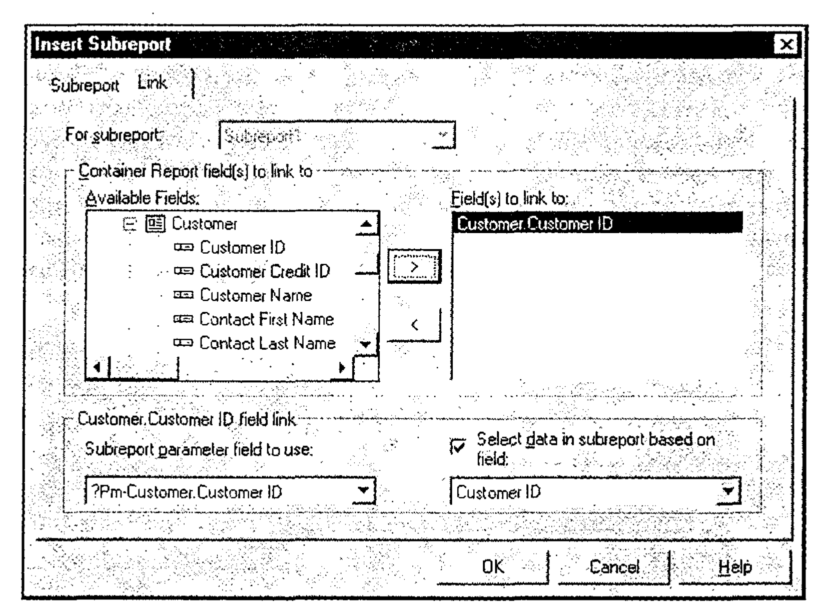


Рис. 6.20. Вкладка **Links** диалогового окна **Insert Subreport**

Следовательно, при связывании отчета и подотчета данные подотчета фильтруются так, чтобы они соответствовали данным основного отчета.

Для связывания можно использовать параметр, созданный в подотчете вручную *(разд. 2.5)*. Все параметры подотчета доступны в списке выбора **Subreport parameter field to use**.

Отчет и подотчет могут быть связаны и посредством формулы. Например, если поле отчета, по которому должно произойти связывание, — число (в примере — **Customer.Customer ID**), а поле подотчета — строка, то в отчете можно создать формулу

```
ToText ({Customer.Customer ID},0 )
```

и переместить ее в список **Field(s) to link to**.

В основном отчете подотчет является перемещаемым и редактируемым объектом. Для редактирования подотчета Crystal Reports 9 создает дополнительную вкладку, соответствующую вкладке **Design** основного отчета, и имя вкладки соответствует имени подотчета. Подотчет может быть отредактирован независимо от основного отчета, в частности опция **Set Datasource Location**, выбираемая командой меню **Database | Set Datasource Location**, в основном отчете и в каждом подотчете имеет собственное значение.

Для форматирования подотчета следует в основном отчете установить на нем указатель, щелкнуть правой кнопкой мыши и выполнить соответствующую команду контекстного меню:

- **Save Subreport As** — сохранение подотчета в отдельном файле rpt;

- **Change Subreport Links** — изменение связывания подотчета;
- **Edit Subreport** — переход в режим редактирования шаблона подотчета;
- **Format Subreport** — вызов диалогового окна **Format Editor** для изменения визуальных свойств подотчета.

Упражнение 6.6

Используя в качестве источника данных таблицу **Customer**, создайте отчет, содержащий диаграмму типа **Bar**, показывающий продажи из поля **Last Year's Sales** по всем штатам (регионам, поле **Region**). В отчет включите только регионы Соединенных Штатов. Включите в основной отчет подотчет, содержащий диаграмму типа **Pie**, показывающий продажи в пяти крупнейших штатах.

Для создания такого отчета необходимо:

1. В основном отчете установить фильтр

   ```
   Country is equal to "USA";
   ```

2. В основном отчете выполнить группирование по полю **Region**;
3. В секцию **Report Header** основного отчета вставить диаграмму типа **Bar**, на вкладке **Data** выбрать тип **Group**;
4. Скрыть все секции основного отчета, кроме **Report Header** и **Report Footer**;
5. В секцию **Report Footer** основного отчета вставить подотчет;
6. В подотчете установить фильтр

   ```
   Country is equal to "USA";
   ```

7. В подотчете установить группировку Top N по полю **Region**;
8. В секцию **Report Header** подотчета вставить диаграмму типа **Pie** (на вкладке **Data** выбрать тип **Group**);
9. Скрыть все секции подотчета, кроме **Report Header**.

Основной отчет и подотчет не должны быть связаны.

Данные могут быть переданы из основного отчета в подотчет и возвращены из подотчета с помощью переменных типа Shared. Для этого отчет и подотчет должны содержать формулы с одной и той же переменной типа Shared.

Упражнение 6.7

Рассмотрим пример передачи параметра из подотчета в основной отчет.

1. Используя в качестве источника данных таблицу **Customer**, создайте отчет, включите в него поля **Customer Name** и **Customer ID**. Создайте множественные секции **Details a** и **Details b**.

2. В секцию **Details b** внесите формулу Shared2:

   ```
   WhilePrintingRecords;
   Shared currencyVar NV1;
   ```

3. Используя в качестве источника данных таблицу **Orders**, создайте подотчет и включите в него поле **Orders.Customer ID**. Свяжите отчет и подотчет по полям **Customer. Customer ID** и **Orders. Customer ID**

4. В секцию **Report Footer** подотчета внесите формулу **Shared1**

   ```
   Shared currencyVar NV1 ;
    NV1 := Sum ({Orders.Order Amount})
   ```

5. Скройте все секции подотчета, кроме **Report Footer**.

6. Вставьте подотчет в секцию **Details a**.

Обратите внимание, что формула Shared2, использующая значение переменной из подотчета, расположена в секции, следующей за секцией, включающей подотчет. Подотчет должен выполняться ранее, чем вычисляется значение формулы, иначе будет передаваться неверное значение переменной NV1.

6.7. Отчет на основе OLAP-источников данных

Эффективный анализ корпоративной информации является сложной задачей. Она усложняется еще и тем, что структура промышленных баз данных, как правило, оптимизируется для оперативной работы с данными и к данным предъявляются требования непротиворечивости, целостности и отсутствия коллизий при выполнении операций редактирования — удаления и вставки данных. К сожалению, обеспечение этих требований путем нормализации схемы базы данных приводит к падению производительности при выполнении аналитических отчетов, а это затрудняет анализ информации. Возможным решением проблемы является создание специализированных баз данных, дополняющих оперативные реляционные. Базы данных, поддерживающие технологию оперативной аналитической обработки данных (Online Analytical Processing, OLAP), представляют собой один из типов таких специализированных баз данных. Технология OLAP является гибкой методикой анализа корпоративных данных. Crystal Reports 9 поддерживает работу с некоторыми OLAP-источниками данных и позволяет создавать отчеты на основе информации, хранящейся в этих базах данных.

Обычно в отчетах Crystal Reports 9 или электронных таблицах данные изображаются в двумерном виде. По строкам и столбцам указывается, например, наименование товара и регион, а в ячейке таблицы приводится сумма продаж конкретного товара в конкретном регионе. С помощью двумерной

таблицы можно исследовать только два измерения, наименование товара и регион в данном примере. Если необходимо исследовать зависимость продаж от периода времени, то придется вводить в отчет дополнительный параметр, например месяц, и использовать его как условие выборки данных. Тогда таблица будет представлять продажи различных товаров в разных регионах, например, в январе. Если нужно рассмотреть, как продавались товары в феврале, необходимо вновь заполнить отчет данными с новым значением параметра, что требует времени. Для анализа того, как продавались товары в одном регионе, но в разные периоды времени, необходимо создать новый отчет и заполнить его данными, что потребует дополнительных ресурсов. В OLAP-источнике данные логически представляются в виде многомерного куба (рис. 6.21).

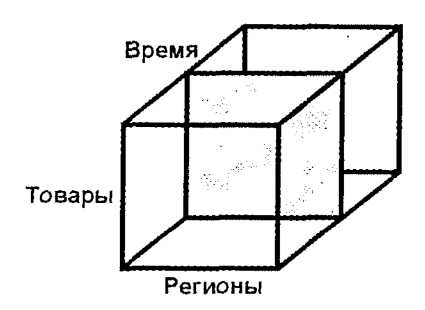


Рис. 6.21. Пример представления данных в виде трехмерного куба

OLAP-сервер позволяет представить данные в виде m-мерного среза *n*-мерного куба с очень высокой производительностью. На рис. 6.21 показан двумерный срез трехмерного куба. Срез представляет продажи товаров в различных регионах, отфильтрованные по определенному периоду времени. OLAP-сервер обеспечивает высокую производительность при выполнении таких операций, как параллельный перенос среза — изменение периода времени, изменение плоскости среза — определение зависимости продажи товара от времени в конкретном регионе или продажи конкретного товара в регионах в разные периоды времени.

Crystal Reports 9 позволяет создавать отчеты на основе данных из OLAP-кубов различных промышленных стандартов, а также комбинировать отчеты, созданные на основе данных из реляционных баз и из OLAP-кубов, например:

- создать стандартный отчет с использованием реляционной базы данных;
- создать OLAP-отчет, с использованием данных исключительно из куба;
- создать стандартный отчет с использованием реляционной базы данных и добавить к нему OLAP-отчет, основанный на кубе;
- создать несколько различных OLAP-отчетов с использованием одного или нескольких кубов.

Отчеты Crystal Reports 9 на основе OLAP-кубов содержат специальный объект — **OLAP grid**, который по структуре аналогичен объекту **Cross-Tab** матричного отчета.

Создать OLAP-отчет можно, выбрав в диалоговом окне **Crystal Reports Gallery** (рис. 2.1) пункт **OLAP**. Появляется мастер отчетов **OLAP Report Creation Wizard**, который содержит несколько последовательно открывающихся окон, позволяющих шаг за шагом создать отчет.

Первое окно мастера **OLAP Data** (рис. 6.22) предназначено для выбора источника данных для отчета. Предварительно должно быть установлено клиентское программное обеспечение выбранного OLAP-сервера.

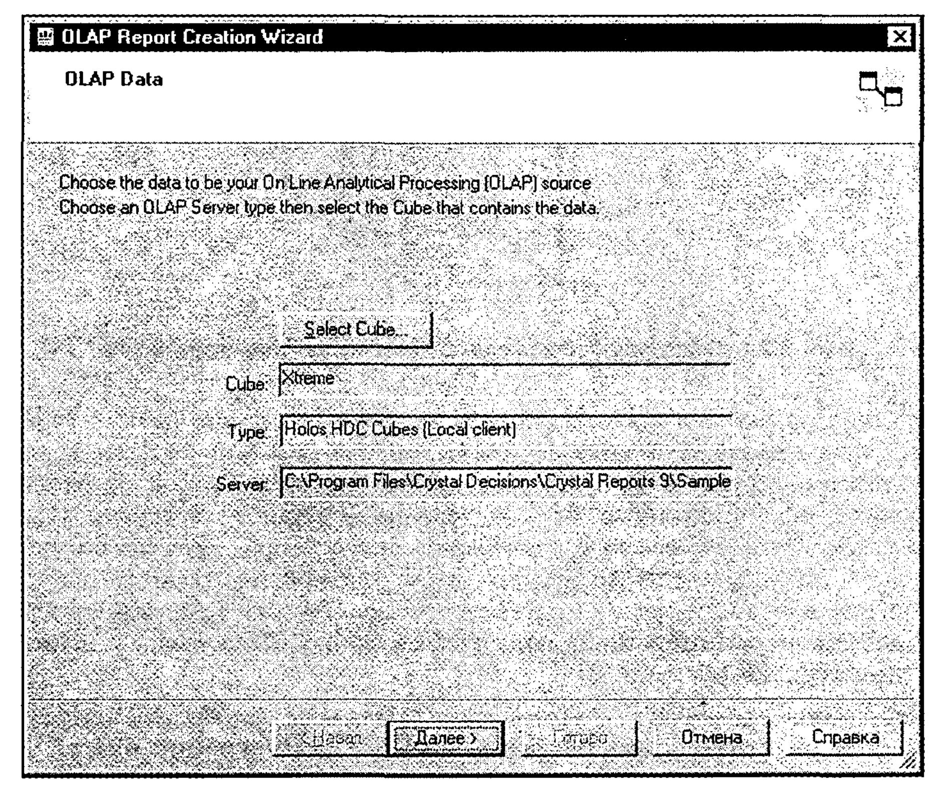


Рис. 6.22. Окно **OLAP Data** мастера **OLAP Report Creation Wizard**

Источником данных может быть OLAP-сервер или файл CAR, созданный в среде **Crystal Analysis Professional**. Для выбора файла CAR следует щелкнуть на кнопке **Select CAR File**. Для выбора в качестве источника данных OLAP-сервера следует щелкнуть на кнопке **Select Cube**. Появляется диалоговое окно **The Crystal OLAP Connection Browser** (рис. 6.23).

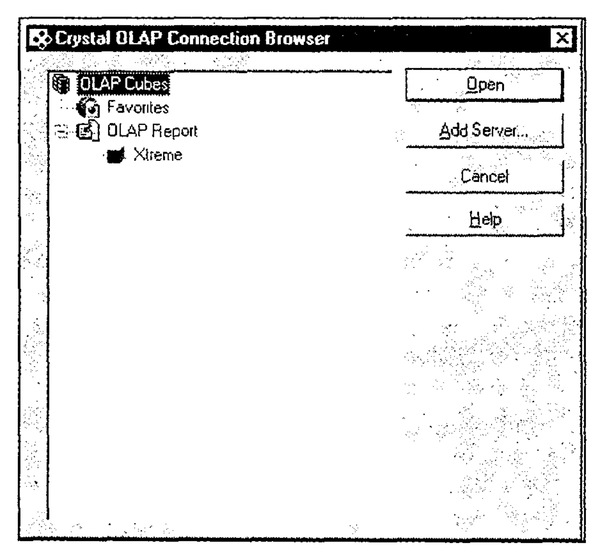


Рис. 6.23. Диалоговое окно **Crystal OLAP Connection Browser**

Рис. 6.24. Диалоговое окно **New Server**

Для создания нового источника данных следует щелкнуть на кнопке **Add Server**. Появляется диалоговое окно **New Server** (рис. 6.24), позволяющее

выбрать опции сеанса связи с OLAP-сервером. В верхней части диалогового окна находится раскрывающийся список **Server Type** с типами связи с сервером. Для большинства серверов используется Microsoft OLE DB Provider for OLAP Services. Для сервера Holos Live Server, опция **Holos HDS Cubes**, используется Open Component Architecture for OLAP (OCA). Кроме того, возможен прямой доступ к серверам IBM DB2 OLAP Server и Hyperion Essbase. В поле **Caption** указывается имя сервера, который будет использоваться для отчета, это поле обязательно для заполнения. Содержимое нижней части диалогового окна — **Server Options** — зависит от выбранного типа связи. Для всех типов кроме Holos Live Server следует указать имя сервера (**Server**), имя пользователя (**User Name**) и пароль (**Password**). Для Holos Live Server необходимо указать имя файла типа HDS. После щелчка на кнопке **OK** диалоговое окно **New Server** закрывается и в списке окна **Crystal OLAP Connection Browser** появляется новый OLAP-источник данных (рис. 6.23). Если в выбранной базе данных содержится более одного куба, то в раскрывающемся списке следует выбрать куб, по которому будет создаваться отчет. Для выбора источника следует переместить указатель на строку в списке OLAP-источников данных и щелкнуть на кнопке **Open**.

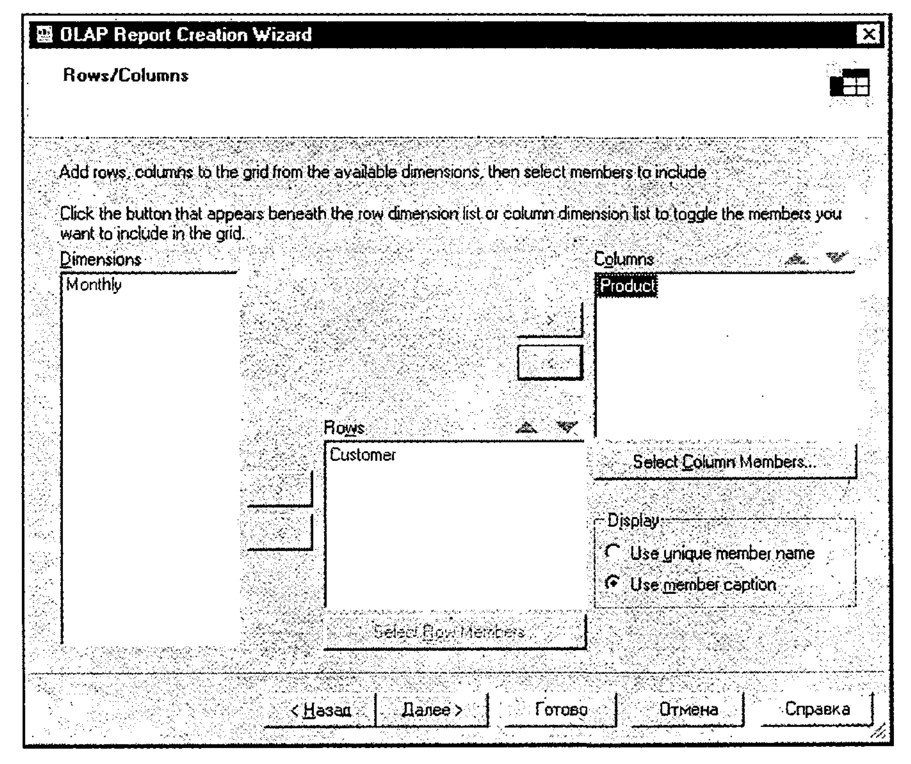


Рис. 6.25. Окно **Rows/Columns** мастера **OLAP Report Creation Wizard**

Щелчок на кнопке **Next (Далее)** открывает второе окно мастера **OLAP Report Creation Wizard-Rows/Columns** (рис. 6.25), позволяющее выбрать измерение или измерения куба, которые будут использоваться в качестве строк и столбцов отчета.

Доступные измерения куба показываются в списке **Dimensions**. Чтобы использовать измерение в качестве строки или столбца, необходимо переместить указатель на строку в списке **Dimensions** и щелкнуть на кнопке > слева от списка **Columns** или **Rows**. В списки **Rows** и **Columns** можно добавлять более одного измерения, в этом случае они будут сгруппированы иерархически, так же как группируются множественные строки и колонки в матричном отчете. Для построения иерархии измерения должны быть перечислены в правильном порядке, например, сначала страна, затем — регион и потом — город. Если размерности внесены в списки **Rows** или **Columns** в неверном порядке, то можно просто перетащить их на правильные позиции, либо выделить измерение и переместить его на правильное положение с помощью клавиш "вверх" и "вниз". Для удаления измерения необходимо переместить указатель на строку в списке **Rows** или **Columns** на соответствующее измерение и щелкнуть на кнопке <.

Измерения куба, на основе которого создается отчет, могут составлять много уровней, содержащих детальную информацию более низкого уровня и образующих иерархию данных в измерении. По умолчанию Crystal Reports 9 показывает только самый верхний уровень иерархии. Для включения других уровней следует щелкнуть на кнопке **Select Row Members** или **Select Column Members**. Появляется диалоговое окно **Member Selector** (рис. 6.26) с древовидным списком значений измерения. Для включения значения на любом уровне иерархии необходимо выбрать его в списке. Если в списке выбрано хотя бы одно значение, в отчет будет включен соответствующий уровень. Для облегчения работы со списком, который может быть весьма обширным, в верхней части диалогового окна **Member Selector** находится раскрывающийся список **Select**, содержащий набор действий со списком значений. Например, можно выбрать все значения списка (**Select All Members**) или отменить выбор (**Select None**), инвертировать выбор (**Invert Selection**), выбрать значение верхнего (**Add Parent to Selection**) или нижнего (**Add Children to Selection**) уровня и т. д.

Третье окно мастера **OLAP Report Creation Wizard — Slice/Page** (рис. 6.27) позволяет включить в отчет те измерения куба, которые не отображаются в качестве строк или столбцов. В левом списке окна **Slice** показываются измерения, не выбранные в качестве строк или столбцов, в правом (**Page**) — измерения, на основе которых отчет будет разбит на группы. Щелчок на кнопке **Select Slice** вызывает диалоговое окно **Member Selector**, где можно указать одно значение измерения, которое войдет в отчет. Данные, соответствующие прочим значениям, в отчет не войдут, другими словами, здесь с помощью **Member Selector** можно установить фильтр по измерениям куба, не выбранным в качестве строк или столбцов.

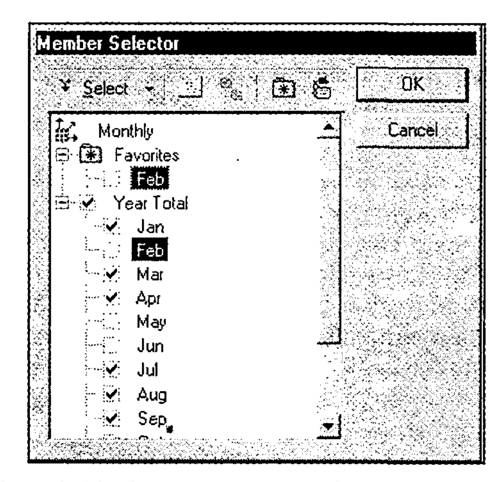


Рис. 6.26. Диалоговое окно **Member Selector**

OLAP Report Creation Wizard

Slice/Page

Choose what to do with the dimensions that are not rows or columns in the grid.
Every grid displayed will have the dimensions set to the value selected for it in the Slice list.
A separate grid will be displayed for each selected value of dimensions in the Page list.

Slice

Monthly = Year Total

Select one value for each dimension.

Select Slice...

Link to Parameter: Create/Edit... Delete

Page

Drag a dimension into this list box and select multiple values.

Select Page Values...

Link to Parameter: Create/Edit... Delete

< Назад | Далее > | Готово | Отмена | Справка

Рис. 6.27. Окно **Slice/Page** мастера **OLAP Report Creation Wizard**

Список **Page** позволяет организовать группирование по измерениям куба. Для внесения измерения в список **Page**, необходимо переместить указатель

на строку в списке **Slice** и щелкнуть на кнопке > слева от списка **Page**. Появляется диалоговое окно **Member Selector** (справа на рис. 6.28). В списке окна **Member Selector** показывается иерархия уровней выбранного измерения. По умолчанию в отчет включается только один верхний уровень. При этом в отчете будет существовать только один объект **OLAP grid**. Для включения нижних уровней следует раскрыть иерархический список и отметить в нем значения, например, на рис. 6.28 для измерения **Monthly** отмечен уровень 0 (Year Total) и несколько значений уровня 1 (Jan, Feb, Mar и Apr). В этом случае в отчете выполняется группирование, во внешнюю группу включается объект **OLAP grid** уровня 0, во внутреннюю группу — объект **OLAP grid** уровня 1, причем во внутренней группе будут показаны только те **OLAP grid** уровня 1, которые соответствуют выбранным в списке окна **Member Selector** значениям. На рис. 6.28 показан вид такого отчета в режиме **Design** в центре и **Preview** справа. Диалоговое окно **Member Selector** можно вызвать для изменения выбранных значений измерения, щелкнув на кнопке **Select Page Values** в окне **Slice/Page** мастера **OLAP Report Creation Wizard**.

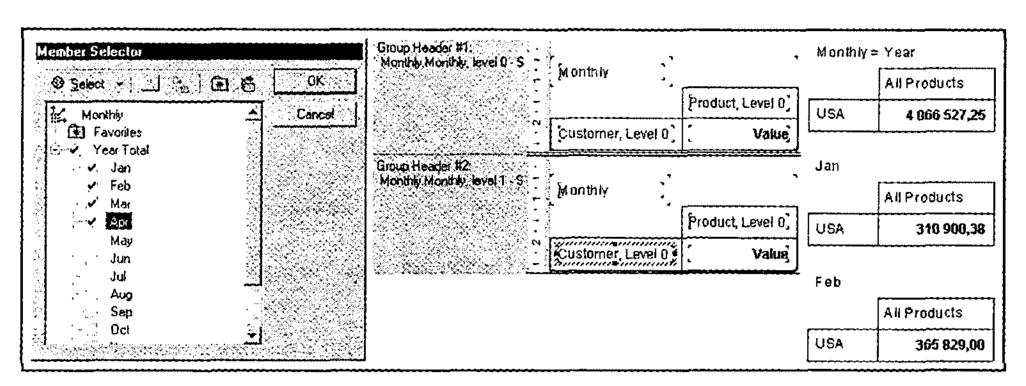


Рис. 6.28. Группирование данных в отчете на основе OLAP-источников данных

Кнопки **Create/Edit** и **Delete** в группе **Link to Parameter** позволяют создавать и удалять параметры для выборки и группировки по измерениям куба. Щелчок на кнопке **Create/Edit** вызывает диалоговое окно **Create Parameter Field** (рис. 2.21), позволяющее создать новый параметр. Создание и использование параметров было рассмотрено в *разд. 2.5.*

Окна **Chart** и **Style** мастера **OLAP Report Creation Wizard** аналогичны окнам **Chart** и **Grid Style** мастера **Cross-Tab Report Creation Wizard** (рис. 6.10 и 6.11) Они предназначены для создания диаграммы на основе данных объекта **OLAP grid** и выбора одного из предопределенных стилей этого объекта. Щелчок на кнопке **Finish (Готово)** закрывает мастер **OLAP Report Creation Wizard** и открывает вновь созданный отчет.

Создать отчет на основе OLAP-источников данных можно, также вставив в стандартный отчет объект **OLAP-grid** с помощью команды меню **Insert/**

6 Зак. 1021

OLAP grid или кнопки панели инструментов для вставки объектов в отчет, см. табл. 1.3. Появляется диалоговое окно **OLAP Expert** (рис. 6.29). Это же окно используется и для редактирования созданного объекта **OLAP-grid**. Чтобы инициировать редактирование, следует щелкнуть на левом верхнем углу **OLAP-grid** и в контекстном меню выполнить команду **OLAP Expert**.

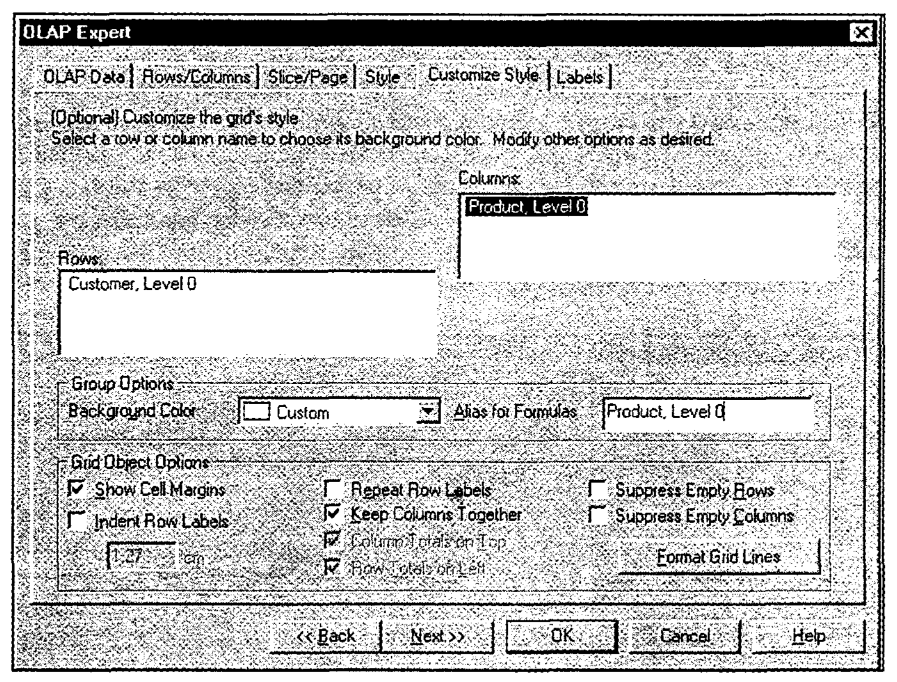


Рис. 6.29. Вкладка **Customize Style** диалогового окна **OLAP Expert**

Вкладки **OLAP Data**, **Rows/Columns**, **Slice/Page** и **Style** диалогового окна **OLAP Expert** аналогичны окнам **OLAP Data**, **Rows/Columns**, **Slice/Page** и **Style** мастера **OLAP Report Creation Wizard** (см. рис. 6.22, 6.25 и 6.27).

Вкладка **OLAP Data** позволяет выбрать OLAP-источник данных. Кнопка **Select Cube** вызывает диалоговое окно **Crystal OLAP Connection Browser** (рис. 6.23), в котором следует указать OLAP-источник данных. Вкладка **Rows/Columns** позволяет выбрать измерение или измерения куба, которые будут использоваться в качестве строк и столбцов объекта **OLAP-grid**. Вкладка **Slice/Page** позволяет выбрать и сгруппировать данные отчета.

На вкладке **Labels** (рис. 6.30) можно выбрать метки к объектам **OLAP-grid** из значений измерений, которые не были выбраны ранее в качестве строк или столбцов.

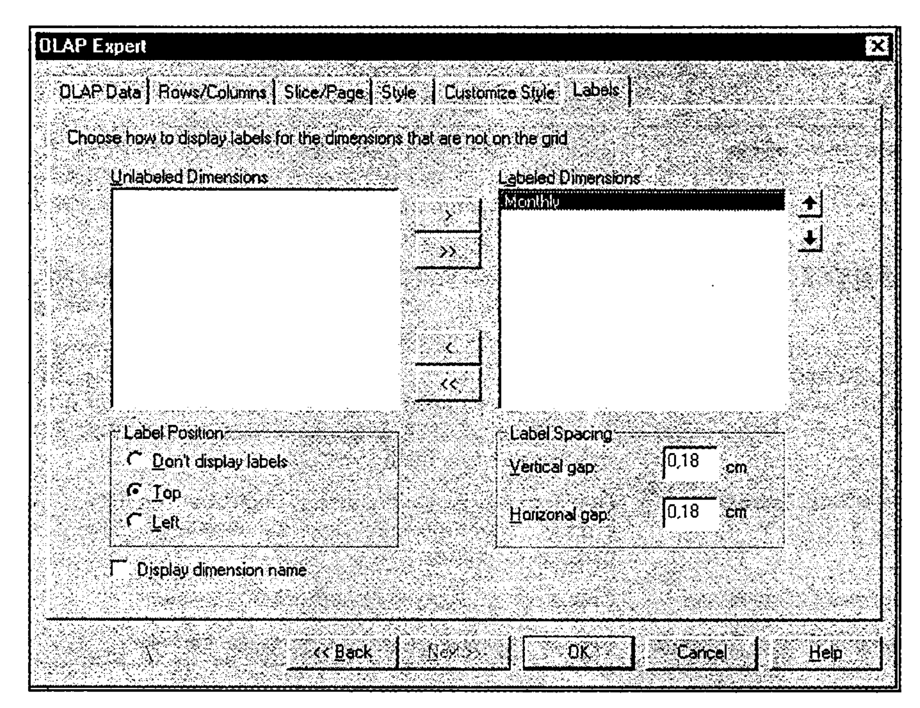


Рис. 6.30. Вкладка **Labels** диалогового окна **OLAP Expert**

Вкладка **Customize Style** (рис. 6.29) позволяет настроить формат создаваемых объектов **OLAP-grid**. Эта вкладка аналогична соответствующей вкладке редактирования матричного отчета. В верхней части вкладки **Customize Style** расположены окна **Rows** и **Columns**. Одновременно можно выбрать только один объект (строку или столбец объекта **OLAP-grid**) в одном из двух окон. Именно для этого объекта можно установить свойства с помощью элементов управления, расположенных в нижней части вкладки. В табл. 6.4 приведены опции форматирования строк и столбцов объекта **OLAP-grid**, доступные на вкладке **Customize Style**.

Таблица 6.4. *Опции форматирования строк и колонок* ***OLAP-grid***

Опция	Описание опции
Alias for Formulas	Используется для ссылки на целую строку или столбец при выполнении условного форматирования в объекте **OLAP-grid**
Background Color	Устанавливает цвет фона для строки или столбца
Show Cell Margins	Окружает ячейки свободным пространством со всех сторон. Отключение этой опции позволяет размещать ячейки более компактно

Таблица 6.4 (окончание)

Опция	Описание опции
Indent Row Labels	Выбор этой опции приводит к отступу подписи (Label) выбранной строки от левого края объекта **OLAP-grid**. Величина отступа строки задается в поле снизу от окна выбора **Indent Row Labels**
Format Grid Lines	Вызывает диалоговое окно **Format Grid Lines**, которое предназначено для задания формата линий сетки в объекте **OLAP-grid**
Keep Columns Together	Защищает столбцы от разбиения, когда распечатываемая таблица превосходит по ширине страницу и таблица объекта **OLAP-grid** должна быть распечатана на двух или более страницах
Repeat Row Labels	При выборе **Keep Columns Together** повторяет метки строки, если ширина объекта **OLAP-grid** превышает ширину страницы
Suppress Empty Rows	Скрывает строки, не содержащие данных
Suppress Empty Columns	Скрывает столбцы, не содержащие данных

Для форматирования готового объекта **OLAP grid** следует, щелкнув в левой верхней части правой кнопкой мыши, выполнить команду контекстного меню **Format OLAP grid**.

После внесения в секцию отчета, на вкладке **Design** объект **OLAP grid** выглядит как набор дочерних полей. Каждое дочернее поле может быть отформатировано индивидуально, в том числе по условию, как обычное поле отчета. Для форматирования поля следует щелкнуть по нему правой кнопкой мыши и выполнить команду контекстного меню **Format Field**. Размер каждого дочернего объекта можно изменить так же, как обычного поля. Можно выделить несколько объектов, а затем отформатировать их все одновременно. Для условного форматирования используется диалоговое окно **Highlighting Expert**, вызываемое выполнением команды **Highlighting Expert** контекстного меню, или с помощью формулы. Формула создается щелчком на кнопке условного форматирования в диалоге **Format Editor**.

Данные, хранящиеся в OLAP-базе данных, можно использовать в основном отчете. Для этого в списке источников данных нужно выбрать источник ODBC или OLE DB в разделе **Create New Connection** списка **Available Data Sources** окна **Data** мастера **Standard Report Creation Wizard**.

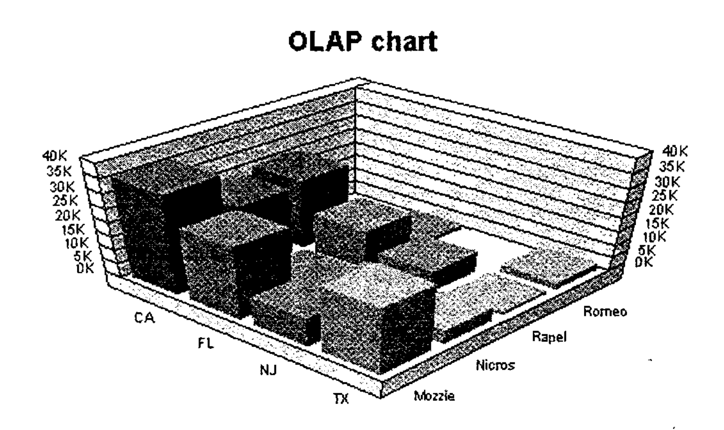
Иногда возникает необходимость переопределить источник данных уже созданного объекта **OLAP-grid**. Для этого необходимо выбрать пункт меню **Database/Set OLAP Cube Location**. В появляющемся диалоге **Set OLAP Cube Location** следует выбрать новый куб OLAP.

Упражнение 6.8

Создайте отчет на основе OLAP-куба.

Для создания такого отчета необходимо:

1. Щелкнуть на кнопке создания нового отчета.
2. Выбрать в диалоговом окне **Crystal Reports Gallery** (рис. 2.1) пункт OLAP.
3. В окне **OLAP Data** мастера **OLAP Report Creation Wizard** щелкнуть на кнопке **Select Cube**.
4. В диалоговом окне **Crystal OLAP Connection Browser** (рис. 6.23) щелкнуть на кнопке **Add Server**.
5. В диалоговом окне **New Server** внести имя источника в поле **Caption**, в качестве типа источника данных выбрать Holos HDC Cube, щелкнуть на кнопке справа от поля HDC File и найти файл Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Databases\Olap Data\ XTREME.HDC, затем щелкнуть на кнопке **OK**.



Year Total

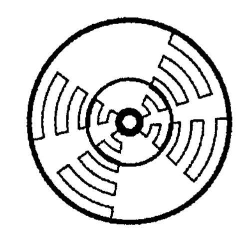
	All Products				
		Mozzie	Nicros	Rapel	Romeo
CA	**401 785,03**	**35 537,83**	**23 533,60**	**24 809,20**	
FL	**165 841,83**	**26 173,96**	**2 687,31**	**14 477,66**	**1 702,60**
NJ	**88 054,79**	**10 400,94**	**6 927,05**	**7 731,84**	
TX	**82 581,82**	**23 941,54**	**5 119,20**	**1 312,11**	**2 497,05**

Рис. 6.31. Пример OLAP-отчета с диаграммой

6. В диалоговом окне **Crystal OLAP Connection Browser** переместить указатель на строку в списке OLAP-источников данных и щелкнуть на кнопке **Open**.
7. В окне **Rows/Columns** мастера **OLAP Report Creation Wizard** внести в качестве столбца **Product** и в качестве строк **Customer**.
8. Щелкнув на кнопке **Select Rows (Columns) Members** в диалоговом окне **Member Selector**, для размерностей **Product** выбрать значения Mozzie, Nicros, Rapel, Romeo. Для размерности **Customer** выбрать значения NJ, TX, FL и CA.
9. Щелкнуть на кнопке **Finish** (Готово).
10. Щелкнуть в левом верхнем углу объекта **OLAP-grid** и выполнить команду меню **Insert Chart**.
11. На вкладке **Type** диалогового окна **Chart Expert** выбрать тип отчета **3D Riser** и на вкладке **Data** установить On Change of = Product, Level 1 и Subdivided by = Customer Level 1.

Полученная диаграмма показана на рис. 6.31.

Глава 7



Связь с базами данных

7.1. Связывание таблиц

Большинство отчетов формируется из данных, находящихся в различных таблицах. Для удобства разработчиков в Crystal Reports 9 включен мастер, который позволяет задать параметры связывания таблиц. В случае использования какого-либо эксперта, позволяющего создать отчет, при выборе нескольких таблиц в диалоговом окне **Data** (Данные) автоматически на следующем этапе открывается окно **Link** (Связывание), показанное на рис. 7.1. При работе с уже созданным отчетом ту же самую функцию по настройке параметров работы с таблицами базы данных выполняет мастер под названием **Database Expert** (Эксперт базы данных), который можно вызвать, нажав на кнопку . Того же самого результата можно добиться при использовании команды меню **Database | Database Expert**. Элементы диалогового окна **Database Expert** (Эксперт базы данных) показаны на рис. 7.2 и 7.3.

В зависимости от того, какой тип источника данных используется при формировании отчета, внешний вид таблиц, отображенных в окне **Link** или на вкладке **Links** в окне **Database Expert**, может незначительно измениться. При работе с настольными базами данных, связь с которыми осуществляется напрямую, некоторые колонки в таблицах помечены цветовыми метками так, как это показано на рис. 7.4. Цветовые метки выделяют проиндексированные колонки. Цвет метки определяет тип индекса. Легенда цветов для проиндексированных колонок показана на рис. 7.5.

При работе с реляционными источниками данных или в случае использования ODBC, ADO, DAO и т. п. для подключения к базе данных, выделение цветовыми метками проиндексированных колонок не происходит. Данный вариант отображен на рис. 7.3.

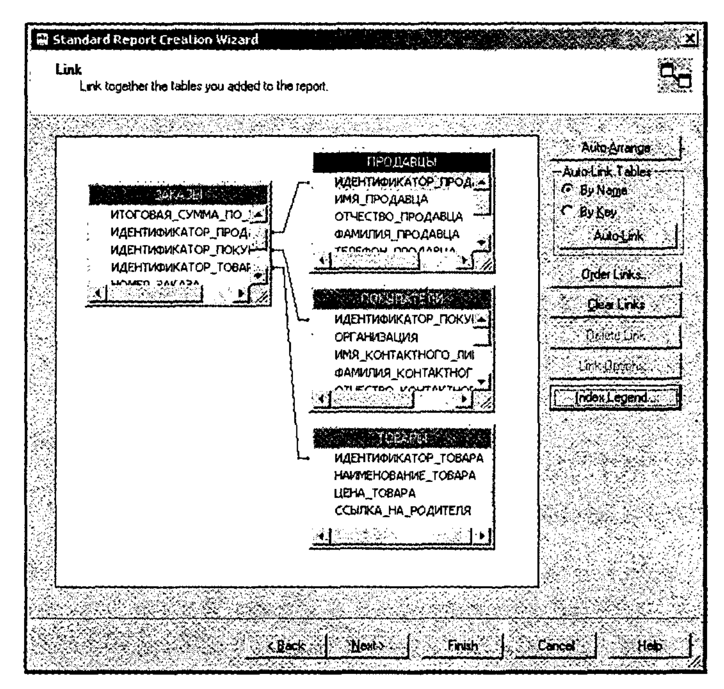


Рис. 7.1. Диалоговое окно **Link**, отображаемое в момент построения отчета с помощью какого-либо эксперта

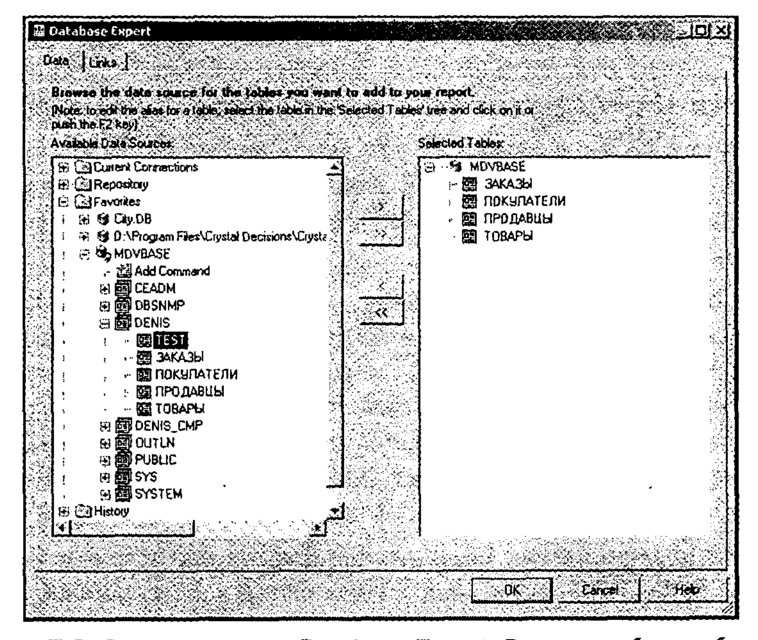


Рис. 7.2. Диалоговое окно **Database Expert**. Вкладка выбора таблиц, используемых для формирования отчета

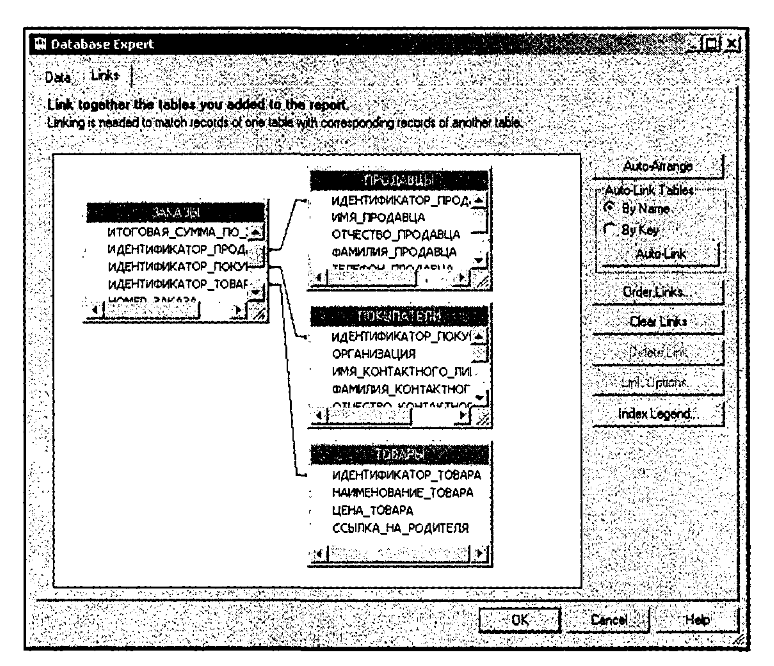


Рис. 7.3. Диалоговое окно **Database Expert**. Вкладка настройки взаимосвязи между таблицами

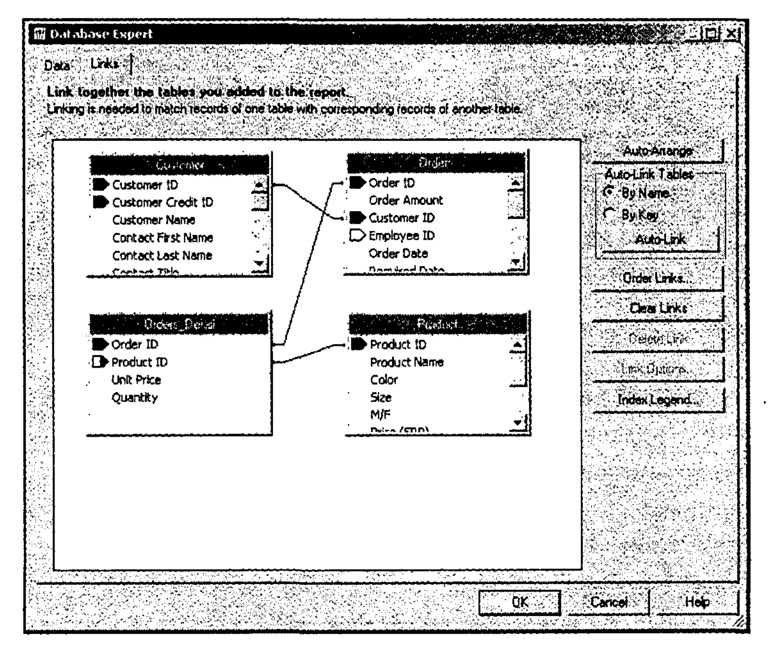


Рис. 7.4. Диалоговое окно **Database Expert**. Вкладка настройки взаимосвязи между таблицами из настольного источника данных

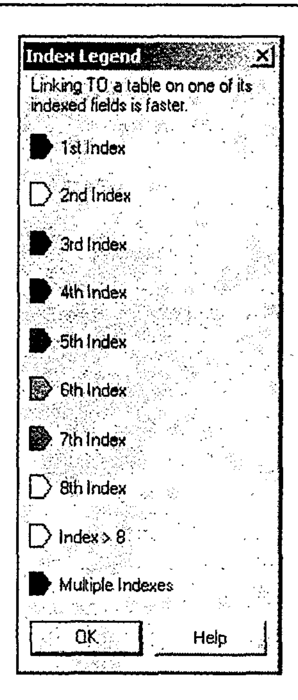


Рис. 7.5. Легенда цветовых меток, используемых при отображении проиндексированных колонок



Crystal Reports версии 9.X отличается от более ранних версий своими возможностями для работы с базами данных. В данной версии генератора отчетов принцип работы с различными типами источников данных абсолютно одинаков. В то же время Crystal Reports версий 8.5 и ниже различает типы источников данных и требует выполнения определенных условий при связывании таблиц. Подробнее о требованиях, предъявляемых к связыванию таблиц в Crystal Reports 8.X, можно узнать в книге "Введение в Crystal Reports", Маклаков С., Матвеев Д. Интерфейс-Пресс. Богородский печатник.

Окно **Link** и вкладка **Links** в окне **Database Expert** имеет следующие элементы управления:

- **Auto-Arrange** — выравнивание таблиц;
- **Auto-Link Tables** — группа элементов, позволяющих выполнить автоматическое связывание таблиц по выбранному условию:
  - **By Name** — связать таблицы по одноименным колонка;
  - **By Key** — связать таблицы по ключевым колонкам;
  - **Auto-Link** — выполнить операцию связывания;

- **Order-Links** — вызывает диалоговое окно (см. рис. 7.6), позволяющее настроить порядок следования связей;
- **Clear Links** — удаление всех связей, установленных автоматически или вручную;
- **Delete Link** — удаление выбранной связи;
- **Link Options** — просмотр свойств связи между таблицами. Диалоговое окно, позволяющее просмотреть и настроить характеристики связи между таблицами, показано на рис. 7.7;
- **Index Legend** — вызов легенды цветовых меток, используемых при отображении индексированных колонок. Легенда показана на рис. 7.5.

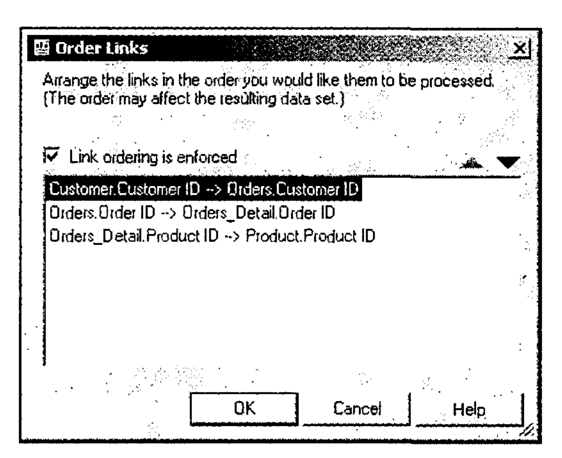


Рис. 7.6. Диалоговое окно **Order Links**, позволяющее установить порядок следования связей

В этом окне настроек имеется возможность установки предписанного порядка следования связей. Для этого необходимо установить флаг **Link Ordering is enforced** (Предопределенный порядок связывания). Также существует возможность изменения заданного порядка. Это достигается нажатием на кнопки ▲▼ после установки фокуса на соответствующую связь.

В данном окне отображена информация о связи в формате `<Имя таблицы №1>.<Имя ключевой колонки>` → `<Имя таблицы №2>.<Имя ключевой колонки>` и две группы параметров, позволяющие изменить установки, заданные по умолчанию, на те, которые требуются для более правильной обработки данных в отчете. В последующих разделах этой главы все параметры данного окна будут рассмотрены подробно.

Кроме кнопок имеется возможность выполнения настроек с помощью контекстного меню, которое вызывается щелчком правой клавиши мыши на поле в таблице или на связи. Для поля будет вызвано меню, показанное на рис. 7.8, а для связи будет показано меню, представленное на рис. 7.9.

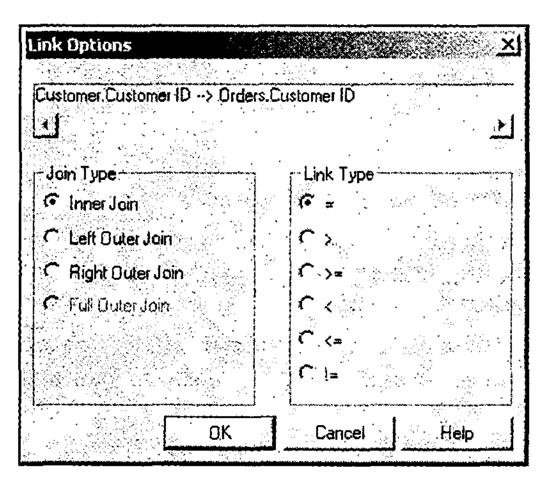


Рис. 7.7. Окно **Link Options**, позволяющее просмотреть и настроить параметры связи между таблицами

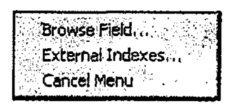


Рис. 7.8. Контекстное меню, вызываемое для выбранного поля в таблице

Команды этого меню предназначены для выполнения следующих задач:

- **Browse Field** — просмотр данных, сохраненных в поле;
- **External Indexes** — вызов диалогового окна, которое позволяет добавить внешние индексы, пункт меню доступен только для dBase;
- **Cancel Menu** — закрытие контекстного меню.

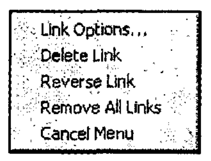


Рис. 7.9. Контекстное меню, вызываемое для выбранной связи между таблицами

Пункты данного меню частично дублируют, а частично дополняют функциональность, предоставляемую кнопками:

- **Link Options** — просмотр свойств связи между таблицами. Диалоговое окно, позволяющее просмотреть и настроить характеристики связи между таблицами, показано на рис. 7.7;

- **Delete Link** — удаление выбранной связи;
- **Reverse Link** — изменение статуса связываемых таблиц. Дочерняя таблица трансформируется в родительскую и наоборот. Данная операция влияет только на работу отчета, на уровне связей в базе данных ничего не меняется;
- **Remove All Links** — удаление всех связей в окне **Link** или на вкладке **Links**;
- **Cancel Menu** — закрытие контекстного меню.

Упражнение 7.1

1. Создайте отчет, использующий в качестве источника данных несколько таблиц, и посмотрите результаты связывания в окне **Link** или **Database Expert**.
2. Посмотрите свойства связей, используя меню и двойной щелчок мыши.

7.2. Доступ к источникам данных

Способы доступа к данным в Crystal Reports 9 можно разбить на пять категорий:

1. Прямой доступ к данным.
2. Доступ с использованием ODBC.
3. Доступ с использованием OLE DB.
4. Доступ посредством файлов Crystal SQL Designer.
5. Доступ посредством файлов Crystal Dictionary.

Для доступа к каждому типу данных используется определенный набор библиотек и другие, необходимые для доступа файлы.

В текущей версии Crystal Reports предусмотрен прямой доступ к следующим источникам данных:

- Access 7, 8, 2000;
- ACT! 3, 4;
- Btrieve;
- Clipper;
- DB2;
- dBASE III+, IV, для Windows и Visual dBase;
- Exchange/Outlook;
- FoxPro 2.5 и более ранние версии;

- Informix;
- Local File System;
- Lotus Domino;
- Microsoft IIS/Proxy Log File;
- NT Event Log;
- Oracle 7, 8.X, 9.X;
- Paradox;
- Sybase Adaptive Server;
- Web Log Files.

Совместно с Crystal Reports 9 поставляется набор ODBC-драйверов, которые можно выбрать в момент установки пакета и использовать в дальнейшем для работы. Кроме того, все ODBC-драйверы, которые были установлены на компьютер ранее или потом, можно также использовать в Crystal Reports 9.

К поставляемым совместно с Crystal Reports 9 библиотекам относятся:

- Required Runtime Files;
- Access 7.0 and Above;
- ASCII Text;
- DB2/2;
- Excel;
- Informix;
- Lotus Domino;
- Microsoft SQL Server;
- Oracle;
- PeopleSoft;
- Sybase Adaptive Server;
- Visual FoxPro;
- XML.

Наличие механизма OLE DB позволяет обратиться к любым источникам данных, для которых есть OLE DB-провайдер или ODBC-драйвер.

Настроить источник ODBC можно с помощью стандартных средств операционной системы. Учитывая то, что Crystal Reports 9 является 32-разрядной программой и работает под управлением Windows 9X, Windows NT или Windows 2000, программа настройки ODBC для всех возможных вариантов одна и та же. Для вызова программы администрирования ODBC-драйверов можно в командном режиме запустить программу ODBCAD32.EXE или же, открыв окно **Контрольная панель (Control Panel)**, выбрать пиктограмму

ODBC-источники данных (ODBC Data Sources). Используя возможности **Администратора источников данных ODBC**, можно добавить и настроить любое количество источников, необходимых для работы. Подробнее о настройке конкретного ODBC-соединения можно прочитать в HELP-файлах, устанавливаемых совместно с драйвером. Также информацию можно получить в справке по ODBC, находящейся в составе HELP по Windows.

Механизмы использования файлов Crystal SQL Designer и Crystal Dictionary предусматривают наличие файлов, созданных с помощью компонент Crystal SQL Designer и Crystal Dictionary, которые входят в Crystal Reports, версий 8.X и младше. Подробное описание процесса создания таких файлов и способов их использования предоставлено в книге "Введение в Crystal Reports", Маклаков С., Матвеев Д. Интерфейс-Пресс. Богородский печатник.

Упражнение 7.2

1. Создайте отчет, который работает с реляционной базой данных за счет прямого подключения. Например, с Oracle.
2. Настройте ODBC-драйвер с помощью **Администратора источников данных ODBC** и создайте отчет в Crystal Reports 9 с использованием этого драйвера.
3. Создайте отчет, используя какой-либо из механизмов OLE DB.

7.3. Настройка параметров Crystal Reports 9 для работы с базами данных

Помимо параметров, которые можно настроить в окнах **Link** и **Database Expert**, существует также возможность определить ряд характеристик, которые будут использованы на уровне всех создаваемых отчетов. Для того чтобы настроить эти параметры, необходимо вызвать диалоговое окно **Options** командой меню **File | Options** и перейти на вкладку **Database** так, как это показано на рис. 7.10. Можно также в диалоговом окне **Data**, вызвав щелчком по правой клавише мыши меню, изображенное на рис. 7.11, выбрать пункт **Options**. При этом появится окно **Options**, которое по своим характеристикам совпадает с вкладкой, изображенной на рис. 7.10. Внешний вид окна **Options** представлен на рис. 7.12.

Элементы контроля в данном окне, позволяющие выполнять настройку, разбиты на две группы:

- **Explorer Options** (Опции выборки) — параметры выборки данных;
- **Advanced Options** (Дополнительные опции) — расширенные возможности.

Options

Fields | Fonts | Smart Tag

Layout | Database | Editors | Data Source Defaults | Reporting

Explorer Options

Show:

Tables

Views

System Tables

Synonyms

Stored Procedures

Table name LIKE: (%,_)

Owner LIKE: (%,_)

List Tables and Fields by:

Name

Description

Both

Sorting:

Sort Tables Alphabetically

Sort Fields Alphabetically

Advanced Options

Use Indexes or Server for Speed

Perform Grouping On Server

Database Server is Case-Insensitive

Select Distinct Data for Browsing

Perform Query Asynchronously

Verify On First Refresh

Verify Stored Procedures On First Refresh

Verify When Database Driver Upgraded

OK | Cancel | Help

Рис. 7.10. Диалоговое окно **Options**

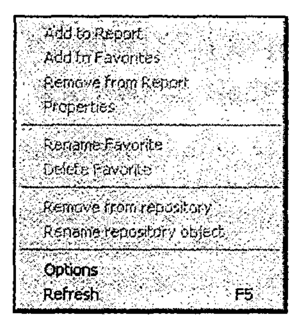


Рис. 7.11. Контекстное меню, вызываемое в окне **Link** или **Database Expert**

Группа **Explorer Options** содержит элементы управления, которые могут обеспечить более удобную работу с базами данных в момент создания отчета. Эта группа разделена на подгруппы:

- **Show**;
- **List Tables and Fields by**;
- **Sorting**.

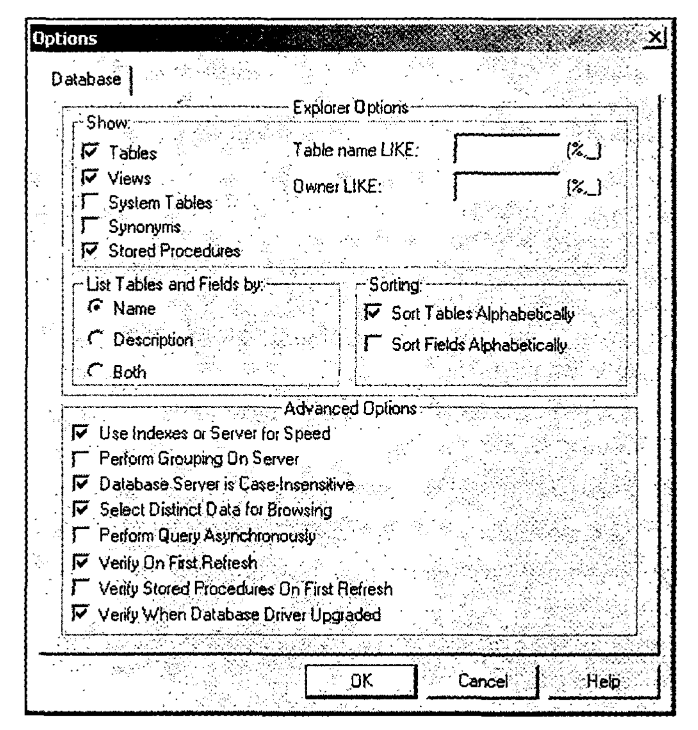


Рис. 7.12. Диалоговое окно **Options**, вызываемое с помощью контекстного меню

Подгруппа **Show** (Показывать) включает в себя следующие элементы контроля:

- **Tables** — если сделать активным данный параметр, то в отчете можно будет использовать данные из таблиц;
- **Views** — если сделать активным данный параметр, то в отчете можно будет использовать данные из представлений;
- **System Tables** — если сделать активным данный параметр, то в отчете можно будет использовать данные из системных таблиц;
- **Synonyms** — если сделать активным данный параметр, то из отчета можно будет обращаться к таблицам по их синонимам, если СУБД поддерживает механизм создания синонимов;
- **Stored Procedures** — если сделать активным данный параметр, то в отчете можно будет использовать хранимые процедуры практически так же, как таблицы;
- **Table name LIKE** — поле, в которое можно вписать имя таблицы или часть имени, используя шаблон, либо перечислить через запятую несколько имен, которые будут использоваться в качестве условия формирования

списка допустимых таблиц. Шаблон имени может формироваться следующим образом — если в поле вписать комбинацию символов CUS%, где знак процента — это любое количество неизвестных символов, то будут выбираться таблицы с именами, начинающимися с CUS и т. п.;

- **Owner LIKE** — если СУБД поддерживает такое понятие, как схема или владелец таблиц, то можно ограничить выбор таблиц, представлений и процедур. Для этого необходимо указать в поле имя владельца либо воспользоваться шаблоном, как в случае параметра **Table name LIKE**. В результате список доступных источников будет ограничен только теми, которые принадлежат заданным владельцам. Имена владельцев желательно указывать в верхнем регистре.

Подгруппа **List Tables and Fields by** (Список таблиц и полей по) включает в себя такие элементы контроля, как:

- **Name** — при работе с данными показывать список имен таблиц и полей;
- **Description** — при работе с данными показывать описание таблиц и полей;
- **Both** — при работе с данными показывать имена и описание таблиц и полей.

Подгруппа **Sorting** (Сортировка) позволяет выполнять сортировку имен таблиц и полей следующим образом:

- **Sort Tables Alphabetically** — сортировать имена таблиц в цифро-алфавитном порядке;
- **Sort Fields Alphabetically** — сортировать имена полей в цифро-алфавитном порядке.

Группа **Advanced Options** содержит набор параметров, которые предоставляют дополнительные возможности, используемые при работе с данными:

- **Use Indexes or Server for Speed** — включает механизм использования индексов или возможности сервера для оптимизации запросов;
- **Perform Grouping on Server** — включает механизм формирования запроса, выполняющего операцию группирования данных на сервере;
- **Database Server is Case-Insensitive** — данный параметр позволяет включить или отключить чувствительность к регистру символов, которые передаются в запросе в качестве условия;
- **Select Distinct Data for Browsing** — включает механизм выборки данных при просмотре без дублирования;
- **Perform Query Asynchronously** — позволяет выполнять запросы к базе данных асинхронно;
- **Verify On First Refresh** — включается механизм проверки структуры используемых данных при выполнении обновления данных в момент вызова отчета на просмотр;

- **Verify Stored Procedures On First Refresh** — включается механизм проверки структуры возвращаемого набора данных при выполнении обновления информации в момент вызова отчета на просмотр;
- **Verify When Database Driver Upgraded** — включается механизм проверки структуры используемых данных в случае изменения библиотеки, используемой для связи с данными.

Упражнение 7.3

Попробуйте менять настройки, предназначенные для работы с базами данных, и посмотрите результаты.

7.4. Использование SQL-запросов при формировании отчетов

В Crystal Reports версии 9 внесены значительные изменения, связанные с использованием SQL-запросов. Как уже было сказано, в более ранних версиях Crystal Reports существовала компонента Crystal SQL Designer, с помощью которой создавались сначала отдельные файлы SQL-запросов, а затем эти файлы подключались к отчету как источник данных. Такой механизм, с одной стороны, позволял разработчику убедиться в правильности работы запроса перед тем, как использовать его в отчете, с другой стороны, имелся ряд значительных ограничений, которые требовалось принимать в расчет. Одним из таких ограничений является то, что Crystal SQL Designer для подключения к базам данных может использовать только ODBC-драйверы. Другое серьезное ограничение заключается в том, что непосредственно при работе с запросом в отчете нельзя его отредактировать для обеспечения требуемого результата. В Crystal Reports версии 9 недочеты более ранних версий устранены, и разработчик обладает большим набором возможностей. Вместо механизма создания отдельного файла с SQL-запросом предоставлена возможность формирования запроса непосредственно в момент выбора источника данных. Для этого, после того как источник данных определен, вместо таблиц, представлений или хранимых процедур можно воспользоваться пунктом **Add Command** (Добавить команду) списка **Available Data Sources** окна **Data**. Данный пункт показан на рис. 7.13. Первая выгода данного механизма заключается в том, что можно сформировать запрос независимо от типа источника, который потом будет использован для построения отчета. Вторая выгода заключается в том, что запрос доступен непосредственно в Crystal Reports, а не через Crystal SQL Designer, и его можно в любой момент исправить. Кроме того, любой запрос, сформированный через раздел **Add Command**, можно сохранить в централизованном хранилище объектов, что дает возможность использовать один и тот же запрос для формирования различных отчетов и, в случае необходимости, использовать этот запрос как прототип для формирования других запросов.

Standard Report Creation Wizard

Data

Choose the data you want to report on.

Available Data Sources:

- Repository
- Favorites
- History
- Create New Connection
  - Access/Excel (DAO)
  - COM Connectivity
  - Crystal Data Object
  - Crystal Queries
    - ...no items found...
  - Database Files
    - Find Database File
    - D:\Program Files\Crystal Decisions
      - Add Command
      - Tables
      - Views
      - Stored Procedures
  - Dictionary/Infoview
  - Field Definitions Only
  - File System Data
  - IDAPI Database DLL
  - Java Beans Connectivity

Selected Tables:

> >> < <<

< Back Next > Finish Cancel Help

Рис. 7.13. Окно **Data**. Команда **Add Command**

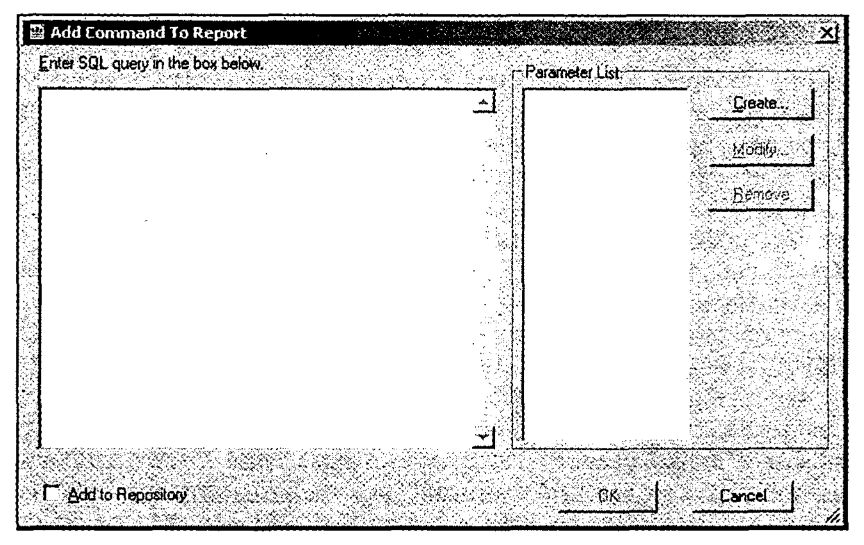


Рис. 7.14. Диалоговое окно **Add Command To Report**, позволяющее оформить новый запрос и задать его параметры

Для того чтобы создать SQL-запрос, необходимо дважды щелкнуть левой клавишей мыши на строке **Add Command**, при этом появится диалоговое окно **Add Command To Report** (Добавить команду в отчет), показанное на рис. 7.14.

Данное окно содержит следующие элементы:

- окно **Enter SQL query in the box below** — в это окно выписывается SQL-запрос или копируется из любого текстового редактора;
- группа **Parameter List** в составе:
  - окно, отображающее список параметров;
  - кнопка **Create** — предоставляет возможность создать параметр для запроса;
  - кнопка **Modify** — дает возможность отредактировать свойства параметра;
  - кнопка **Remove** — позволяет удалить параметр;
- опция **Add to Repository** (Добавить в хранилище) — дает возможность сразу же после формирования запроса записать его в хранилище как самостоятельный объект, пригодный к многоразовому использованию.

Запрос, пригодный к использованию, отображен на рис. 7.15.

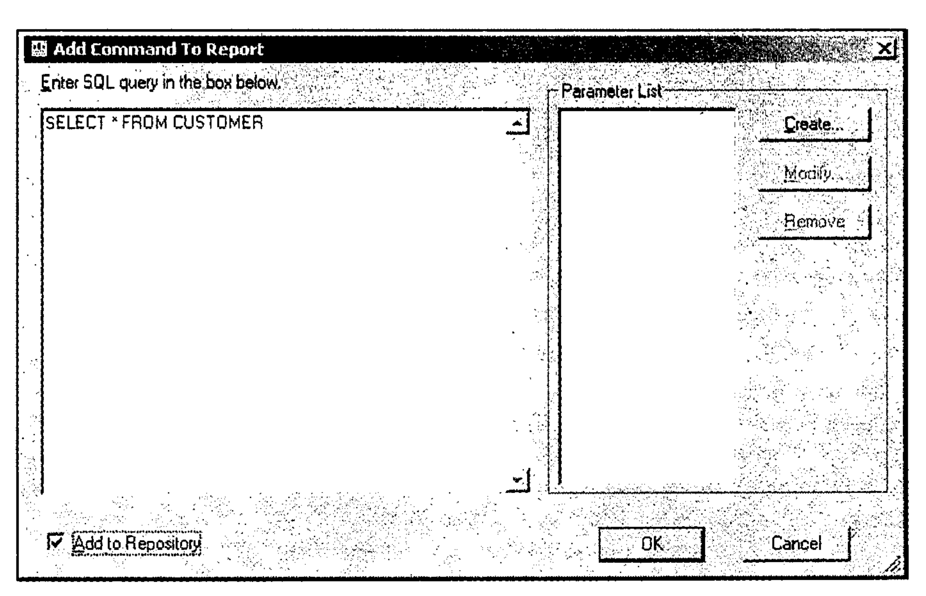


Рис. 7.15. Диалоговое окно **Add Command To Report** с написанным запросом

В случае, когда установлена опция **Add to Repository** так, как показано на рис. 7.15, при нажатии на кнопку **OK** сначала появляется окно **Add Item**

(Добавить элемент), в котором можно дать имя запросу как самостоятельному объекту, уточнить его синтаксис и указать, в какой раздел хранилища будет записан запрос. Окно **Add Item** показано на рис. 7.16.

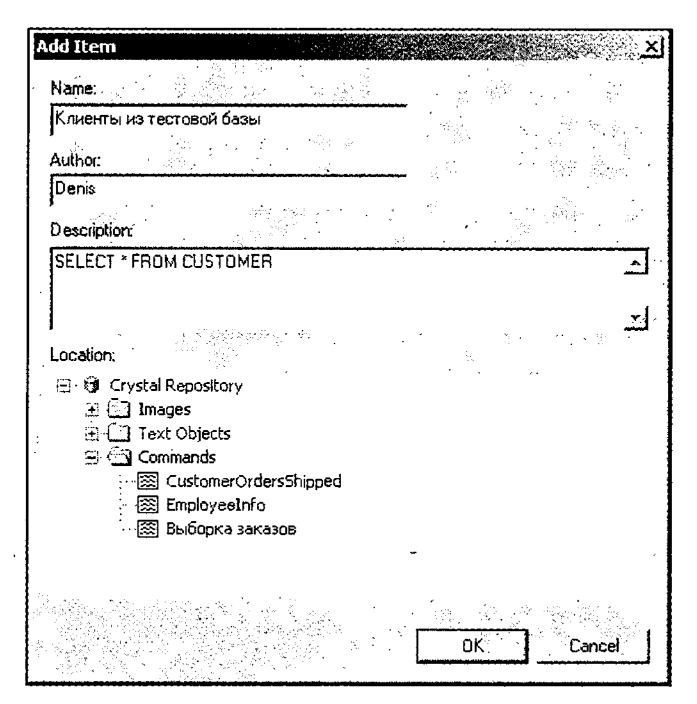


Рис. 7.16. Окно **Add Item**

После того как процесс записи запроса в хранилище завершен, запрос автоматически попадает в раздел **Selected Tables** (Выбранные таблицы) окна **Data** так, как это показано на рис. 7.17.

Все последующие действия по формированию отчета совпадают с механизмами, описанными в предыдущих главах данной книги.

Как уже было определено ранее, любой SQL-запрос может содержать один или несколько параметров, которые будут обрабатываться на уровне отчета так же, как и поля-параметры. Для того чтобы создать параметр, который затем может быть использован в запросе, необходимо нажать на кнопку **Create** (Создать), при этом откроется окно **Command Parameter** (Параметр SQL-команды), показанное на рис. 7.18, позволяющее определить свойства параметра.

Механизмы работы с параметрами совпадают с механизмами, которые описаны в *разд. 2.5*. Однако существует ряд отличий, о которых необходимо помнить. Параметр, используемый в SQL-запросе, не может быть оформлен как диапазон или массив. Параметр в запросе всегда определяется как дискретное значение.

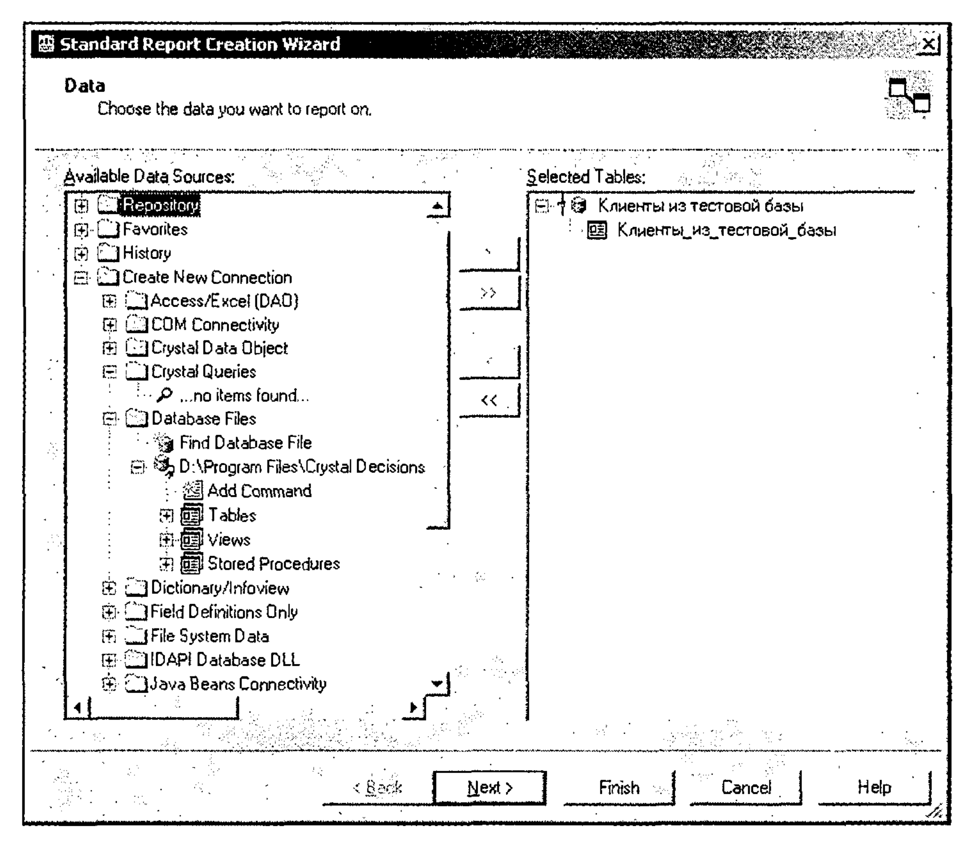


Рис. 7.17. Окно **Data** с выбранным запросом в качестве источника данных

Рис. 7.18. Окно **Command Parameter**, позволяющее определить свойства параметра, используемого в SQL-запросе

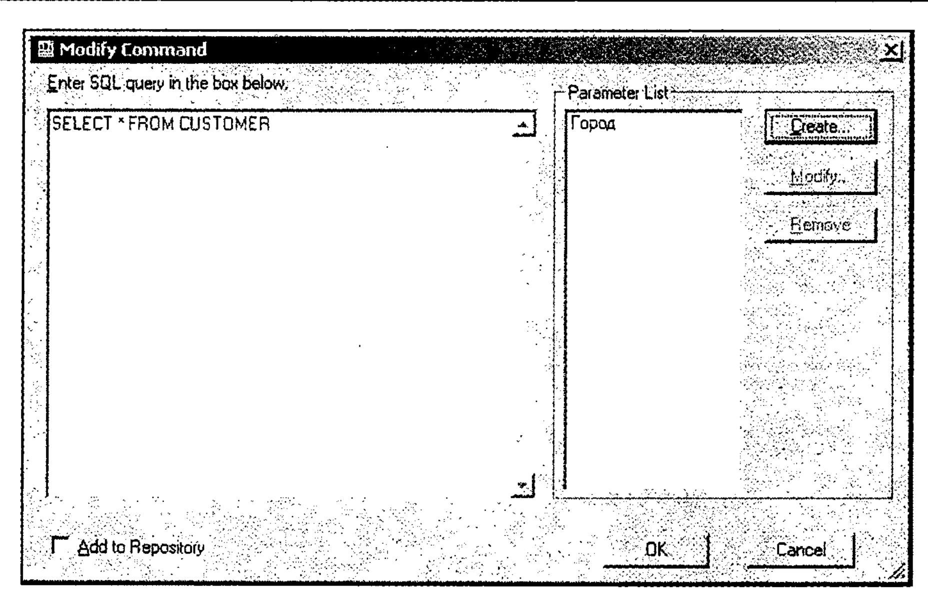


Рис. 7.19. Окно **Modify Command**, содержащее информацию о параметре

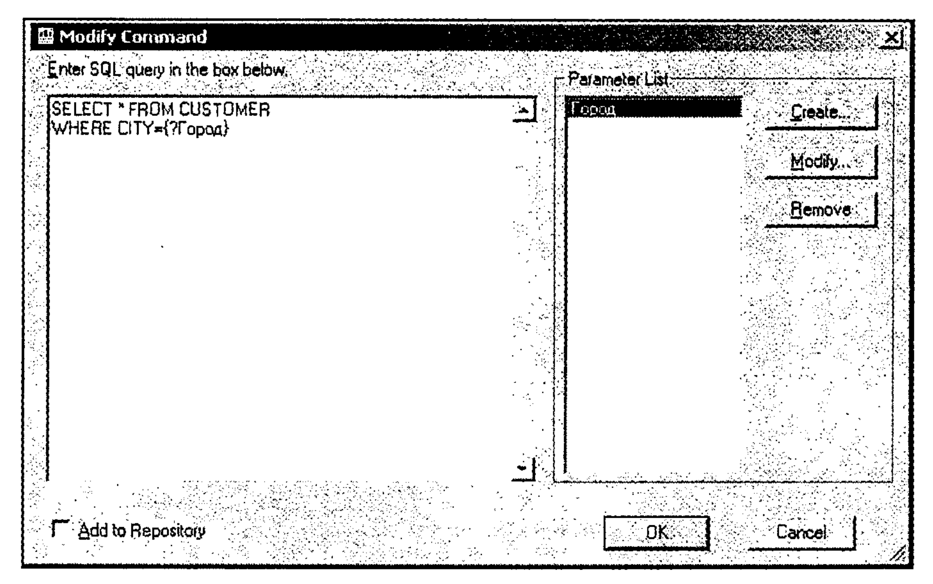


Рис. 7.20. Окно **Modify Command** с запросом, содержащим параметр

Для того чтобы добавить параметр в SQL-запрос, необходимо установить курсор в окне **Enter SQL query in the box below** (Впишите SQL-запрос в поле) в то место, куда требуется поместить параметр, и затем, дважды щелкнув левой клавишей мыши по параметру в списке, переместить его в текст

запроса. При использовании такого механизма добавления параметров в запрос автоматически учитывается необходимый синтаксис. Результат включения параметра в запрос показан на рис. 7.20.

В случае, когда отчет формируется на основании запроса, содержащего параметр, в момент выбора запроса появляется окно, предлагающее задать значение параметра или нескольких параметров. Окно определения значения для параметра показано на рис. 7.21. Точно такое же окно будет отображаться в отчете в случае обновления данных.

Рис. 7.21. Окно **Enter Parameter Values**

В случае, когда возникает потребность в запросе, записанном в хранилище, необходимо использовать раздел **Repository** в окне **Data**. В хранилище выбирается нужный запрос из соответствующего раздела и двойным щелчком левой клавишей мыши или нажатием на кнопку со стрелочкой, направленной вправо, производится подключение запроса как источника данных для отчета. Окно **Data** с разделом **Repository** показано на рис. 7.22.

Если запрос настроен на работу с источником данных, запрашивающим имя пользователя и пароль, в момент подключения появляется окно, предлагающее указать требуемую информацию.

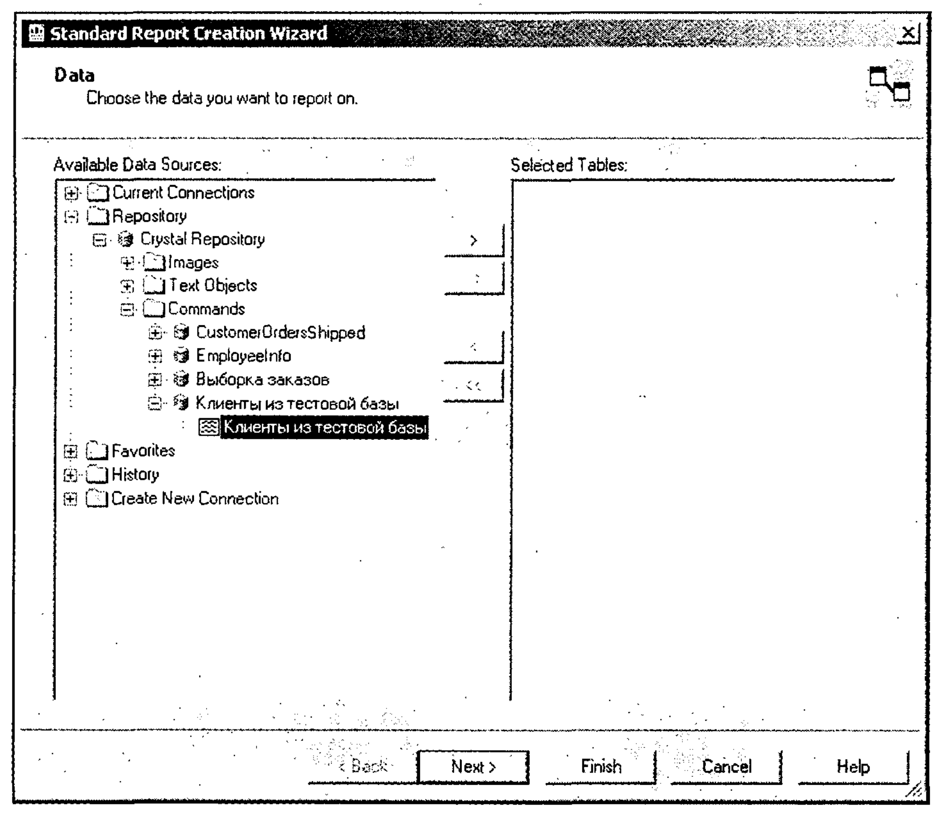


Рис. 7.22. Окно **Data**. Выбор SQL-запроса из хранилища

Если запрос, который содержится в хранилище, используется только в качестве шаблона, требующего изменений, необходимо выполнить следующие действия:

1. Из хранилища выбирается запрос, который может быть использован в качестве шаблона, и подключается как источник данных.
2. Для выбранного запроса правой клавишей мыши вызывается контекстное меню, показанное на рис. 7.23.
3. В контекстном меню выбирается команда **Disconnect from Repository** (Отключиться от хранилища) для того, чтобы освободить запрос.
4. После отключения от хранилища повторно вызывается то же самое контекстное меню, только теперь в нем доступна команда **Edit Command** (Редактировать команду).
5. После выполнения команды меню **Edit Command** появляется окно **Modify Command** (Модификация команды), показанное на рис. 7.19 и 7.20. В этом окне можно выполнять все действия по изменению запроса.

6. Вновь полученный вариант запроса, так же как и новый запрос, можно использовать в пределах одного отчета, а можно записать в хранилище как самостоятельный объект.

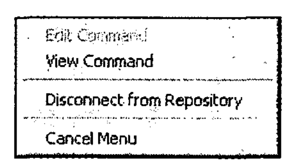


Рис. 7.23. Контекстное меню, позволяющее отключить запрос от хранилища



Одновременное использование SQL-запроса, таблиц, представлений и хранимых процедур недопустимо.

7.5. Введение в язык SQL

Для правильного построения SQL-запросов необходимо знать операторы и синтаксис языка SQL. Учитывая то, что материал этой книги посвящен использованию Crystal Reports, рассмотрение языка SQL будет ограничено его аспектами, имеющими приложение непосредственно к генератору отчетов.

Если рассматривать язык SQL в полном объеме, можно разбить его операторы на три основных категории:

1. Язык определения данных или **Data Definition Language** — создание объектов базы данных.
2. Язык манипулирования данными или **Data Manipulation Language** — определяет, какие данные представлены в таблицах.
3. Язык управления данными или **Data Control Language** — определяет, может ли пользователь выполнить действие.

При работе Crystal Reports 9 используется только язык манипулирования данными или категория Data Manipulation Language. К данному разделу относятся операторы:

- INSERT — вставка данных;
- UPDATE — модификация данных;
- DELETE — удаление данных;
- SELECT — выборка данных.

Учитывая то, что Crystal Reports 9 не изменяет данные в источниках, операторы INSERT, UPDATE и DELETE рассматриваться не будут. Будет

рассмотрен синтаксис запросов, обеспечивающих выборку данных и их комбинацию. В качестве источника данных можно пользоваться любой базой, которая есть у вас под рукой. Если нет ни одного источника данных, можно взять базу данных Microsoft Access (xtreme.mdb), которая устанавливается совместно с Crystal Reports 9 и находится в каталоге примеров. Использование этой базы данных описано в первых главах данной книги.

Все SQL-запросы состоят из предложений или clauses. В качестве примера предложений можно представить следующие конструкции:

- FROM Customer
- WHERE CUSTOMER.CUSTOMERID=ORD.CUSTOMERID

Базовая конструкция SQL-запроса для выборки данных из одной таблицы представлена на схеме 7.1.

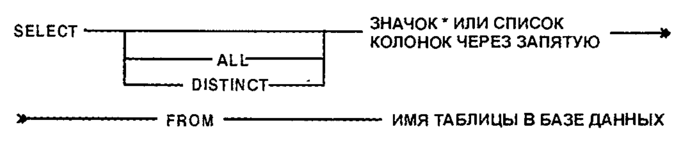


Схема 7.1. Базовая конструкция запроса типа Select

- Предложение ALL указывает на необходимость выборки всех данных из таблицы. Это предложение не является обязательным.
- Предложение DISTINCT позволяет выбирать данные из таблицы без повтора дублирующихся строк.
- Значок звездочка указывает на то, что необходимо выводить данные из всех полей таблицы.

Упражнение 7.4

1. Напишите запросы, выбирающие данные из разных таблиц, и сформируйте на их основании отчеты.
2. Сравните результаты, которые получаются при использовании предложений ALL и DISTINCT.

В процессе работы с запросами может оказаться, что набор данных, которые возвращаются выражением, слишком велик. Для того чтобы сократить количество строк, можно в запросе установить фильтр. Способ установки фильтра показан на схеме 7.2.

При задании параметров фильтрации можно использовать различные реляционные операторы:

- = — равенство;

- > — больше;
- < — меньше;
- >= — больше или равно;
- <= — меньше или равно;
- <>, != — не равно и др.

SELECT — ALL / DISTINCT — ЗНАЧОК * ИЛИ СПИСОК КОЛОНОК ЧЕРЕЗ ЗАПЯТУЮ →

→ FROM — ИМЯ ТАБЛИЦЫ В БАЗЕ ДАННЫХ — WHERE — УСЛОВИЕ ФИЛЬТРАЦИИ ДАННЫХ

Схема 7.2. Конструкция запроса SELECT с установленным фильтром



Оператор "не равно" в различных СУБД может быть определен с помощью различных символов.

Запрос с установленным фильтром может выглядеть так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE COUNTRY='Russia'
```

Кроме реляционных операторов существует набор булевых операторов, которые могут быть использованы совместно с реляционными:

- AND — логическое и;
- OR — логическое или;
- NOT — логическое отрицание.

Примеры совместного использования реляционных и булевых операторов:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE COUNTRY='Russia' AND SUMM>1000
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE COUNTRY='Russia' OR SUMM>1000
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE COUNTRY='Russia' AND NOT SUMM>1000
```

Используя круглые скобки, можно комбинировать различные булевы операторы и создавать сложные формулы фильтрации данных.

Например:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE NOT (COUNTRY='Russia' OR SUMM>1000)
```

Упражнение 7.5

Напишите запросы с фильтрацией данных, используя все возможные операторы, и посмотрите результаты в виде отчетов.

В ряде случаев может оказаться, что реляционных и булевых операторов недостаточно для реализации запроса, возвращающего удовлетворительный

результат. Для решения специфических проблем существует набор специальных операторов, которые облегчают разработку корректных запросов.

Очень часто возникает необходимость установить фильтрацию не по одному значению, а по целому списку. Реализовать такое ограничение можно с помощью оператора OR. Это будет выглядеть так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE COUNTRY='Russia' OR
COUNTRY='USA' OR COUNTRY='Germany' ...
```

Такое перечисление не всегда удобно использовать, и, кроме того, средства разработки приложений и генераторы отчетов могут иметь ограничение на длину строки. Для того чтобы сделать строку SQL-запроса короче, можно воспользоваться оператором IN. В этом случае строка запроса будет выглядеть так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE COUNTRY IN ('Russia', 'USA',
'Germany', ...)
```

За счет такого определения списка значений можно сократить длину запроса практически вдвое.

При необходимости задать диапазон значений также существует два варианта реализации запроса. Первый вариант — это использование реляционных операторов >= и <=:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE SUMM>=1000 AND SUMM<=5000
```

Второй вариант предполагает использование оператора `BETWEEN ... AND ...`:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE SUMM BETWEEN 1000 AND 5000
```

Так же как и в первом варианте, этот оператор включает границы диапазона. Если необходимо исключить границы диапазона, можно реализовать это так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE SUMM>1000 AND SUMM<5000
```

либо:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE SUMM BETWEEN 1000 AND 5000
AND SUMM NOT IN (1000,5000)
```

Очень часто бывает так, что поля базы данных содержат специальное значение NULL. NULL — это не пробел и не ноль, это отсутствие какого-либо значения. Для его обработки можно использовать специализированные операторы *IS NULL* и *IS NOT NULL*. Запросы, которые определяют наличие в поле значения NULL либо значения любого другого имеющегося в базе данных типа, выглядят так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE PHONE IS NULL;
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE PHONE IS NOT NULL
```

В случае необходимости можно организовать фильтр, который выполняет отбор данных не по целому значению, а по подстроке. К сожалению, этот оператор ориентирован на работу только с текстовыми значениями. Для ра-

боты с подстрокой используется оператор LIKE. Конструкция подстроки, подставляемой в качестве условия, использует следующие специальные символы %, * и ?. Если требуется указать, что строка должна начинаться с символа или набора символов, то условие выглядит так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE CITY LIKE 'K%'
```

здесь отбираются города с названиями, начинающимися на букву К.

Если необходимо выполнить поиск строки, которая завершается заданным набором символов, то условие выглядит так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE CITY LIKE '%K'
```

В случае, когда требуется искать строки, которые содержат комбинацию символов в любом месте, запрос можно построить так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE CITY LIKE '%K%'
```

В этом запросе отбираются значения, содержащие букву К в любом месте.

В ряде случаев может возникнуть потребность осуществлять поиск по подстроке, в которой не все символы известны, тогда для указания неизвестных символов можно использовать оператор подчеркивание. Количество значков подчеркивания соответствует количеству неизвестных символов. Запрос в этой ситуации выглядит так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER WHERE CITY LIKE '%K__z%'
```

В этом запросе отбираются значения, содержащие в любом месте фрагмент из трех букв, начинающийся на К и кончающийся на z с произвольным символом на втором месте.

Упражнение 7.6

1. Используя предложенный вариант базы данных, напишите запрос, выбирающий всех клиентов, имена которых начинаются с буквы К, и постройте отчет.
2. Напишите запрос, выбирающий всех клиентов, в именах которых встречается буква К, и постройте отчет.
3. Напишите запрос, выбирающий всех клиентов, имена которых начинаются с любой буквы от А до К, и постройте отчет.
4. Напишите запрос, выбирающий всех клиентов, у которых не введены сведения о городе, и постройте отчет.

В большинстве случаев запросы необходимы не только для того, чтобы выбрать и отфильтровать данные, но и для того, чтобы выполнить какие-либо действия над данными либо их преобразование в тот вид, который будет удобен для работы. Для работы с данными с помощью SQL-запросов существует набор функций, который в зависимости от выбранной СУБД может меняться, но большинство из них идентичны и их синтаксис один и тот же.

К таким функциям относятся агрегатные или статистические функции. Вот список наиболее часто встречающихся статистических функций:

- AVG — вычисление среднего значения по полю;
- SUM — вычисление суммы по полю;
- COUNT — вычисление количества записей;
- MIN или MAX — вычисление минимального или максимального значения поля.

Кроме этих функций существует еще очень большое количество других, но как уже было сказано, для каждой СУБД этот набор различен, и информацию о них необходимо уточнять в документации, сопровождающей конкретный тип и версию СУБД. Так же некоторые СУБД имеют механизмы, которые позволяют пользователю создавать свои функции, отличные от базовых. Использование таких функций в SQL-запросах может существенно упростить работу по построению отчетов на основе запросов.

Запросы, которые используют описанные функции, могут выглядеть так:

```
SELECT AVG(SUMM) FROM CUSTOMER WHERE CITY='Moscow'
SELECT SUM(SUMM) FROM CUSTOMER WHERE CITY='Moscow'
SELECT MIN(SUMM) FROM CUSTOMER WHERE CITY='Moscow'
SELECT MAX(SUMM) FROM CUSTOMER WHERE CITY='Moscow'
```

Запрос, который будет использовать функцию COUNT, может иметь различные варианты оформления:

- `SELECT COUNT(*) FROM CUSTOMER` — вычисление количества записей в таблице;
- `SELECT COUNT(CITY) FROM CUSTOMER` — вычисление количества записей в таблице, без учета тех, где поле CITY имеет значение NULL;
- `SELECT COUNT(DISTINCT CITY) FROM CUSTOMER` — вычисление количества записей в таблице, считая две записи с одним значением поля CITY за одну и без учета тех, где поле CITY имеет значение NULL.

Примечание

При попытке выполнить запрос, имеющий функцию COUNT(DISTINCT CITY), в Microsoft Access будет выдано сообщение об ошибке.

Необходимо учитывать то, что статистические функции возвращают единственное значение и использование их совместно с реальными колонками приведет к ошибке, если не использовать предложение GROUP BY.

Текст сообщения об ошибке может меняться, но ее смысл не меняется. Базовый синтаксис использования предложения GROUP BY показан на схеме 7.3. Предложение GROUP BY объединяет записи с одинаковыми значениями в указанном списке полей в одну запись.

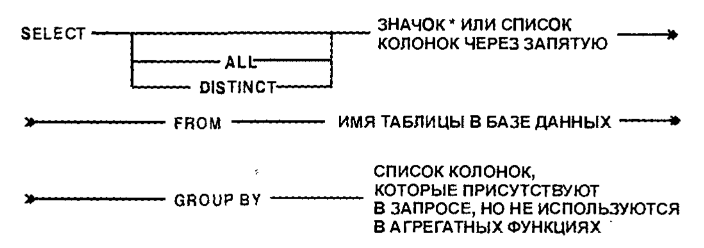


Схема 7.3. Базовая конструкция запроса, использующего предложение GROUP BY

Сам запрос в этом случае будет выглядеть так:

```
SELECT NAME, COUNT(CITY) FROM CUSTOMER GROUP BY NAME
```

Если требуется наложить ограничение на выбираемый набор значений, то можно воспользоваться предложением WHERE. Это предложение должно стоять перед предложением GROUP BY (схема 7.4).

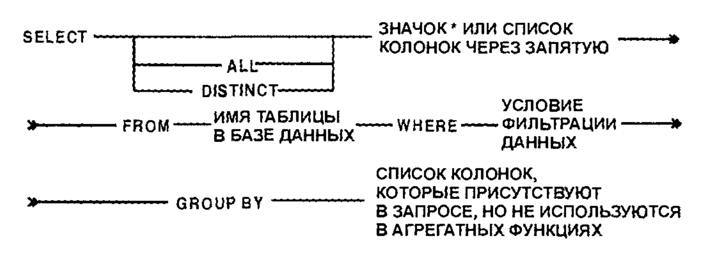


Схема 7.4. Структура запроса, использующего GROUP BY и WHERE

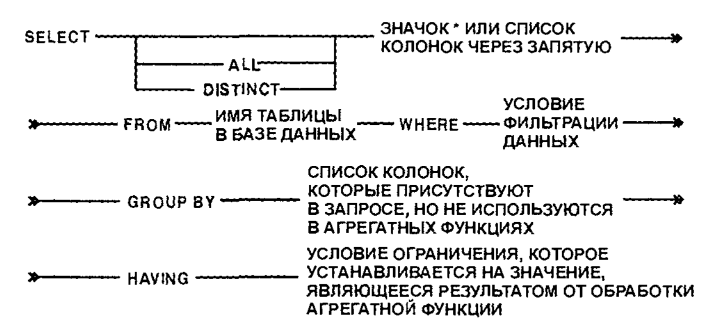


Схема 7.5. Структура запроса, использующего предложение HAVING совместно с GROUP BY

Пример запроса:

```
SELECT NAME, COUNT(CITY) FROM CUSTOMER WHERE SUMM>1000 GROUP BY NAME
```

Если потребуется наложить дополнительные ограничения на выбранные и сгруппированные данные, можно воспользоваться предложением HAVING,

7 Зак. 1021

которое позволяет установить фильтр на значения, возвращаемые агрегатной функцией (схема 7.5). Предложением WHERE в этом случае пользоваться нельзя.

В таком исполнении запрос будет выглядеть так:

```
SELECT NAME, COUNT(CITY) FROM CUSTOMER WHERE SUMM>1000 GROUP BY NAME
HAVING COUNT(CITY)>1
```

Упражнение 7.7

1. Используя имеющуюся у вас структуру данных, напишите запрос, подсчитывающий количество городов в таблице клиентов, и создайте на его основании отчет.
2. Напишите запрос, выбирающий товар с наименьшей ценой, и создайте на его основании отчет.

Результаты, которые возвращаются запросом, не всегда удовлетворяют требованиям форматирования. Для того чтобы результат выглядел красивее, можно в текст запроса включать данные, которые не относятся непосредственно к источнику. При этом запрос может иметь следующий вид:

```
SELECT 'Наименование',NAME, 'Количество', COUNT(CITY) FROM CUSTOMER WHERE
SUMM>1000 GROUP BY NAME HAVING COUNT(CITY)>1
```

При выполнении такого запроса результирующий набор данных будет содержать в первой и третьей колонках константы — "Наименование" и "Количество", а во второй и четвертой колонках два поля из базы данных — NAME и COUNT(CITY) соответственно. При желании можно объединить все поля в одно, для этого используются операторы конкатенации или слияния строк. Для различных СУБД эти операторы могут иметь различный вид. Наиболее часто встречающиеся варианты — это || (две вертикальных черты) или + (плюс). В случае слияния данных из нескольких полей необходимо учитывать, что типы данных должны быть строковыми. Если необходимо объединить данные из строкового поля и цифрового, то требуется преобразовать цифровое значение в строку, хотя некоторые СУБД и не требуют явного преобразования. Предыдущий запрос, использующий слияние данных, можно описать так:

```
SELECT 'Наименование '||NAME||' Количество '||TO_CHAR(COUNT(CITY)) FROM
CUSTOMER WHERE SUMM>1000 GROUP BY NAME HAVING COUNT(CITY)>1
```

Результатом выполнения такого запроса будет всего лишь одна колонка, содержащая объединение данных из нескольких полей базы данных и дополнительных элементов форматирования.

Однако при использовании слияния данных либо функций заголовок результирующего набора данных будет выглядеть некрасиво, причем для различных типов источников эти заголовки будут различаться.

Для того чтобы результат был более красивым и понятным, можно определить псевдоним для тех колонок или сборных данных, которые отображаются некорректно. Псевдоним колонки задается с помощью предложения AS. При этом сам запрос будет выглядеть так:

```
SELECT 'Наименование '||NAME||' Количество '||TO_CHAR(COUNT(CITY)) AS
Информация FROM CUSTOMER WHERE SUMM>1000 GROUP BY NAME HAVING COUNT(CITY)>1
```

Псевдонимы у колонок могут быть использованы также в случае необходимости представления пользователю данных более понятным образом.

Учитывая то, что данные в реляционных базах хранятся в произвольном порядке, возникает необходимость их сортировки. Для сортировки используется предложение ORDER BY. Если необходимо сортировать данные в порядке возрастания, используется ключ ASC либо ничего не указывается, т. е. сортировка по возрастанию применяется по умолчанию. В случае, когда требуется сортировка по убыванию, употребляется ключ DESC. Используя комбинацию предложения ORDER BY и ключей ASC/DESC, можно организовать сортировку данных по разным полям, причем по каждому полю может быть установлен свой порядок. Возможное представление запроса может выглядеть так:

```
SELECT NAME, CITY, SUMM FROM CUSTOMER ORDER BY NAME
SELECT NAME, CITY, SUMM FROM CUSTOMER ORDER BY NAME ASC
SELECT NAME, CITY, SUMM FROM CUSTOMER ORDER BY NAME DESC
SELECT NAME, CITY, SUMM FROM CUSTOMER ORDER BY NAME, CITY
SELECT NAME, CITY, SUMM FROM CUSTOMER ORDER BY NAME ASC, CITY DESC
```



При использовании предложения ORDER BY необходимо учитывать то, что некоторые СУБД позволяют совмещение этого предложения с GROUP BY, а некоторые нет.

Упражнение 7.8

1. Используя имеющуюся структуру базы и набор данных, напишите запрос, выдающий список товаров в порядке убывания стоимости и создайте на его основании отчет.
2. Напишите запрос, который имеет несколько условий сортировки, и создайте на его основании отчет. Измените порядок сортировки, посмотрите результат.

Часто возникает необходимость объединять данные из двух и более таблиц так же, как при построении отчета с помощью редактора **Database Expert**. Такой механизм в языке SQL называется JOIN (соединение). Для объединения данных из двух и более таблиц в запросе необходимо описать условие соединения в предложении WHERE. Базовый способ предполагает равенство ключевых полей в связываемых таблицах. При этом необходимо учитывать

то, что при описании связи необходимо указывать полное имя поля. Полное имя поля, как минимум, состоит из имени таблицы и имени поля:

<Имя таблицы>.<Имя поля>.

Синтаксически конструкция запроса с JOIN формируется так, как это показано на схеме 7.6.

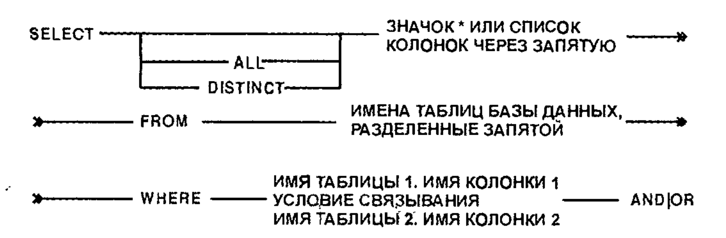


Схема 7.6. Структура запроса с условием соединения таблиц JOIN

Как видно из схемы, в случае, когда таблицы имеют более одного ключевого поля, условия сравнения объединяются через логическое и — AND. То же самое будет происходить при связывании данных из трех и более таблиц. Пример запроса выглядит так:

```
SELECT CUSTOMER.NAME, CUSTOMER.CITY, CUSTOMER.SUMM, ORD.ORDERID,
ORD.DATEO, ORD.DATER, ORD.SUMM FROM CUSTOMER, ORD WHERE
CUSTOMER.CUSTOMERID = ORD.CUSTOMERID;
```

```
SELECT CUSTOMER.NAME, CUSTOMER.CITY, CUSTOMER.SUMM, SALESPEOPLE.NAME,
SALESPEOPLE.CITY, SALESPEOPLE.COUNTRY, SALESPEOPLE.COMM, ORD.ORDERID,
ORD.DATEO, ORD.DATER, ORD.SUMM FROM CUSTOMER, SALESPEOPLE, ORD WHERE
CUSTOMER.CUSTOMERID = ORD.CUSTOMERID AND SALESPEOPLE. SALESPEOPLEID =
ORD. SALESPEOPLEID
```

Если требуется изменить условия связывания данных, то можно воспользоваться любыми другими реляционными операторами вместо равенства.

В ряде случаев может возникнуть ситуация, представленная на рис. 7.24—7.26.

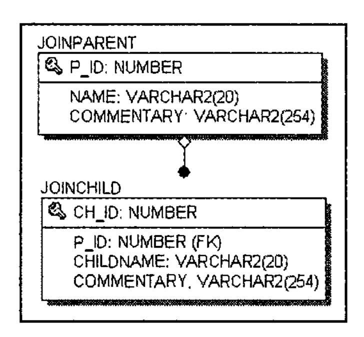


Рис. 7.24. Две связанные таблицы

1,00	IVAN	NO COMMENT
2,00	SVETLANA	NO COMMENT
3,00	VASILIJ	NO COMMENT
4,00	MAIJA	NO COMMENT
5,00	PETR	NO COMMENT

Рис. 7.25. Данные в родительской таблице

11,00	1,00	VANYA	NO COMMENT
12,00	1,00	OLGA	NO COMMENT
13,00	2,00	ALEX	NO COMMENT
14,00	3,00	EGOR	NO COMMENT
15,00	3,00	IRA	NO COMMENT
16,00	4,00	SLAVA	NO COMMENT
17,00		MIHAIL	NO COMMENT
18,00		STEPAN	NO COMMENT

Рис. 7.26. Данные в дочерней таблице

Если обратиться к этим таблицам с запросом:

```
SELECT JOINPARENT.NAME, JOINPARENT.COMMENTARY, JOINCHILD.CHILDNAME,
JOINCHILD.COMMENTARY FROM JOINPARENT, JOINCHILD WHERE JOINPARENT.P_ID =
JOINCHILD.P_ID ,
```

то результат будет выглядеть так, как это представлено на рис. 7.27.

IVAN	NO COMMENT	VANYA	NO COMMENT
IVAN	NO COMMENT	OLGA	NO COMMENT
SVETLANA	NO COMMENT	ALEX	NO COMMENT
VASILIJ	NO COMMENT	EGOR	NO COMMENT
VASILIJ	NO COMMENT	IRA	NO COMMENT
MAIJA	NO COMMENT	SLAVA	NO COMMENT

Рис. 7.27. Выборка данных с условием связывания Equal Join

Как видно из рис. 7.27, записи результирующего набора данных составляются только из тех записей связанных таблиц, которые имеет полное соответствие значений в ключевых колонках, т. е. отбираются только те записи родительской таблицы, которым соответствует хотя бы одна запись дочерней и наоборот. Это не всегда удобно. Иногда надо получить такой набор данных, который содержит не только связанные значения, но и информацию из записей, не связанных с другими. Для этого используются механизмы Left и Right Outer Join. При работе с Oracle такая обработка выполняется с помощью

значка +. Выражение с условием связывания Left Outer Join для СУБД Oracle будет выглядеть так:

```
SELECT JOINPARENT.NAME, JOINPARENT.COMMENTARY, JOINCHILD.CHILDNAME,
JOINCHILD.COMMENTARY FROM JOINPARENT, JOINCHILD WHERE JOINPARENT.P_ID =
JOINCHILD.P_ID(+).
```

Результат работы этого запроса показан на рис. 7.28.

IVAN	NO COMMENT	VANYA	NO COMMENT
IVAN	NO COMMENT	OLGA	NO COMMENT
SVETLANA	NO COMMENT	ALEX	NO COMMENT
VASILIJ	NO COMMENT	EGOR	NO COMMENT
VASILIJ	NO COMMENT	IRA	NO COMMENT
MAIJA	NO COMMENT	SLAVA	NO COMMENT
PETR	NO COMMENT		

Рис. 7.28. Результат при использовании условия связывания Left Outer Join

Как видно из рисунка, количество строк больше, чем при использовании условия Equal Join. При этом данные из родительской таблицы показаны полностью. Из дочерней же таблицы отображены только те, которые имеют полное соответствие ключевых значений. Если использовать условие Right Outer Join, то запрос выглядит так:

```
SELECT JOINPARENT.NAME, JOINPARENT.COMMENTARY, JOINCHILD.CHILDNAME,
JOINCHILD.COMMENTARY FROM JOINPARENT, JOINCHILD WHERE JOINPARENT.P_ID(+)
= JOINCHILD.P_ID
```

IVAN	NO COMMENT	VANYA	NO COMMENT
IVAN	NO COMMENT	OLGA	NO COMMENT
SVETLANA	NO COMMENT	ALEX	NO COMMENT
VASILIJ	NO COMMENT	EGOR	NO COMMENT
VASILIJ	NO COMMENT	IRA	NO COMMENT
MAIJA	NO COMMENT	SLAVA	NO COMMENT
		MIHAIL	NO COMMENT
		STEPAN	NO COMMENT

Рис. 7.29. Результат обработки запроса с использованием условия связывания таблиц Right Outer Join

Кроме символа + в ряде случаев для определения Left и Right Outer Join может быть использован символ *. Синтаксис запроса при этом несколько меняется и выглядит так:

- для Left Outer Join

```
SELECT JOINPARENT.NAME, JOINPARENT.COMMENTARY, JOINCHILD.CHILDNAME,
JOINCHILD.COMMENTARY FROM JOINPARENT, JOINCHILD WHERE JOINPARENT.P_ID*
= JOINCHILD.P_ID
```

- для Right Outer Join

```
SELECT JOINPARENT.NAME, JOINPARENT.COMMENTARY, JOINCHILD.CHILDNAME,
JOINCHILD.COMMENTARY FROM JOINPARENT, JOINCHILD WHERE JOINPARENT.P_ID
= *JOINCHILD.P_ID
```

Для работы с Microsoft Access синтаксис запроса должен значительно отличаться от ранее описанных вариантов. Условие будет выглядеть так:

```
FROM JOINPARENT LEFT OUTER JOIN JOINCHILD ON JOINPARENT.P_ID =
JOINCHILD.P_ID
```

При работе с большим количеством связанных таблиц существует вероятность того, что текст запроса будет очень длинным и, как следствие, трудным для восприятия. Для того чтобы упростить структуру запроса и укоротить его длину, можно воспользоваться псевдонимами таблиц, которые могут быть короткими. Принцип задания псевдонима таблицы таков:

```
FROM JOINPARENT T1, JOINCHILD T2
```

Как следствие, весь запрос будет выглядеть так:

```
SELECT T1.NAME, T1.COMMENTARY, T2.CHILDNAME, T2.COMMENTARY FROM
JOINPARENT T1, JOINCHILD T2 WHERE T1.P_ID = T2.P_ID
```

Кроме возможности сделать запросы более короткими и изящными, псевдонимы позволяют упоминать одну и ту же таблицу в предложении FROM несколько раз. Пример такого запроса:

```
SELECT G1.GOODS, G2.GOODS, G3.GOODS FROM GOODS G1, GOODS G2 GOODS G3
WHERE G1.Price=100 AND G2.Price=500 G3.Price=300
```

Однако при выполнении такого запроса необходимо учитывать то, что результирующий набор данных будет содержать избыточную информацию.

Упражнение 7.9

1. Используя имеющуюся у вас базу и набор данных, напишите запрос, выдающий список заказов с указанием имен заказчика и продавца, и создайте отчет.
2. Напишите запрос, выдающий пары клиентов из одного города, и создайте отчет.
3. Напишите запрос, выдающий пары клиентов из одного города. Повторы типа "Иванов-Петров" — "Петров-Иванов" исключите. Создайте отчет.

Часто возникает необходимость получить данные, условия фильтрации в которых формируются с помощью другого запроса. Для реализации такой обработки можно использовать механизм вложенных SQL-запросов. Одним из вариантов вложенных SQL-запросов является операция сравнения данных в предложении WHERE. При описании сравнения можно использовать ключевые слова:

- ANY;

- ALL;
- SOME.

Для сравнения характеристик данных операторов рассмотрим два запроса.

```
SELECT * FROM GOODS WHERE PRICE*QUANTITY = ANY (SELECT SUMM FROM ORD
WHERE DATEO < TO_DATE('19/01/2001','DD/MM/YYYY'));
```

```
SELECT * FROM GOODS WHERE PRICE*QUANTITY < ALL (SELECT SUMM FROM ORD
WHERE DATEO < TO_DATE('19/01/2001','DD/MM/YYYY'));
```



Ключ SOME используется как синоним ANY.

Результат работы первого запроса будет содержать записи, в которые произведение PRICE*QUANTITY равно какому-нибудь значению поля SUMM в множестве записей, отобранных при выполнении подзапроса. Результатом работы второго запроса станет набор записей, в которые произведение PRICE*QUANTITY меньше любого значения поля SUMM во множестве записей, отобранных при выполнении подзапроса.

В качестве другого варианта вложенных SQL-запросов можно рассмотреть установку фильтрации данных по списку значений, получаемых другим SQL-запросом. Для этого используется оператор IN или NOT IN. Запрос, использующий оператор IN и вложенный запрос, выглядит так:

```
SELECT * FROM GOODS WHERE ORDERID IN (SELECT ORDERID FROM ORD WHERE
DATEO<TO_DATE('19/01/2001','DD/MM/YYYY'))
```

Иногда требуется получить данные из одной таблицы, но при этом проверить наличие связанных данных в дочерней. Простейшим примером может служить такой вариант, когда есть таблица, содержащая информацию о менеджерах и связанная с ней таблица заказов. Необходимо получить информацию о тех менеджерах, которые выписали хотя бы один заказ. Для этого можно воспользоваться обычным запросом с использованием JOIN, но не всегда это удобно, так как может возникнуть избыточность в возвращаемом наборе данных. С помощью предложения EXISTS реализация запроса позволит сделать запрос более качественным. Запрос будет выглядеть так:

```
SELECT NAME FROM SALESPEOPLE WHERE EXISTS (SELECT * FROM ORD WHERE
ORD.SALESPEOPLEID=SALESPEOPLE.SALESPEOPLEID)
```

Упражнение 7.10

1. Создайте отчет на основе запроса, использующий подзапрос, выдающий список заказов, которые принял продавец Иванов.
2. Создайте отчет на основе запроса, выдающего продавцов, сумма заказов которых выше среднего.
3. Создайте отчет на основе запроса, использующего подзапрос, выдающий продавцов, имеющих более одного клиента.

4. Создайте отчет на основе запроса, выдающего клиентов, сумма на счету которых не меньше, чем сумма на счету любого клиента из Москвы.

Кроме возможности написания запросов с подзапросами существует также механизм объединения нескольких запросов в один. Для этого используются предложения UNION или UNION ALL. Структура такой конструкции показана на схеме 7.7.

SELECT ...——→UNION [ALL]——→SELECT ...——→UNION [ALL] ...

Схема 7.7. Структура запроса с UNION

При построении такого запроса необходимо учитывать то, что структура возвращаемого результата формируется первым запросом. Все последующие запросы, после UNION, должны возвращать такое же количество полей, что и первый запрос, и типы данных должны совпадать соответственно. Возможные варианты запросов могут выглядеть так:

```
SELECT NAME FROM SALESPEOPLE WHERE SALESPEOPLE.CITY =
'London' UNION SELECT NAME FROM CUSTOMER WHERE CUSTOMER.CITY =
'London'
```

```
SELECT NAME, CITY FROM SALESPEOPLE WHERE COUNTRY =
'Russia' UNION SELECT NAME, CITY FROM CUSTOMER WHERE COUNTRY = 'Russia'
```

Для сортировки данных в таких запросах можно также использовать предложение ORDER BY со списком полей. В данном случае можно использовать порядковый номер поля из списка SELECT вместо явного указания имени поля, по которому будет выполняться сортировка. Этот механизм поддерживается не всеми СУБД.

```
SELECT NAME, CITY FROM SALESPEOPLE WHERE COUNTRY =
'Russia' UNION SELECT NAME, CITY FROM CUSTOMER WHERE COUNTRY =
'Russia' ORDER BY 2
```

Также за счет использования предложения UNION можно организовать запрос, в котором одна часть исключает строки выбранные в другой.

```
SELECT 'Клиенты из России '||NAME FROM CUSTOMER WHERE COUNTRY =
'Russia' UNION SELECT 'Другие клиенты '||NAME FROM CUSTOMER WHERE COUNTRY !=
'Russia'
```

Упражнение 7.11

Напишите запрос, выдающий клиентов жителей Лондона, сумма на счету которых не меньше 1000, и клиентов жителей Москвы, сумма на счету которых не меньше 500, и постройте на его основе отчет.

7.6. Использование хранимых процедур в Crystal Reports при построении отчетов

Как уже говорилось ранее, в Crystal Reports 9 можно использовать не только таблицы и представления в качестве источников данных, но и хранимые процедуры. Рассмотрим данную возможность на основе СУБД Oracle и создадим набор элементов, часть из которых потом будет использована в Crystal Reports 9. Для начала реализуем таблицу и заполним ее данными.

Создание таблицы

```
CREATE TABLE Test_Table
(ID number(5),
Firstname varchar2(30),
Lastname varchar2(30),
Birthday date);
```

Вставка данных

```
INSERT INTO Test_Table VALUES
(1, 'Christopher', 'Jones', '01-Nov-70');
INSERT INTO Test_Table VALUES
(2, 'Maria', 'Marshall', '02-Jan-77');
INSERT INTO Test_Table VALUES
(3, 'Jonathan', 'Campbell', '09-Aug-75');
INSERT INTO Test_Table VALUES
(4, 'Julie', 'Gagnon', '23-Dec-72');
INSERT INTO Test_Table VALUES
(5, 'Daemon', 'Thompson', '11-Feb-69');
```

Создание пакета, необходимого для реализации процедуры

```
CREATE OR REPLACE PACKAGE Test_Package
AS TYPE Test_Type IS REF CURSOR RETURN Test_Table%ROWTYPE;
END Test_Package;
/
```

Создание процедуры

```
CREATE OR REPLACE PROCEDURE Test_Procedure (
Test_Cursor IN OUT Test_Package.Test_Type,
```

```
Test_Parameter IN Test_Table.ID%TYPE)
AS
BEGIN
OPEN Test_Cursor FOR
SELECT * FROM Test_Table
WHERE Test_Table.ID = Test_Parameter;
END Test_Procedure;
/
```

После создания описанных объектов базы данных можно перейти к работе с Crystal Reports 9 и настроить параметры среды на работу с хранимыми процедурами, как описано в *разд. 7.3.* Далее можно приступать к созданию отчета с использованием хранимых процедур. Так как Crystal Reports 9 позволяет устанавливать соединение с СУБД Oracle и с помощью ODBC и напрямую, выберите наиболее удобный для вас вариант. На рис. 7.30 показано диалоговое окно **Data** (Данные) из которого осуществлено подключение к Oracle напрямую.

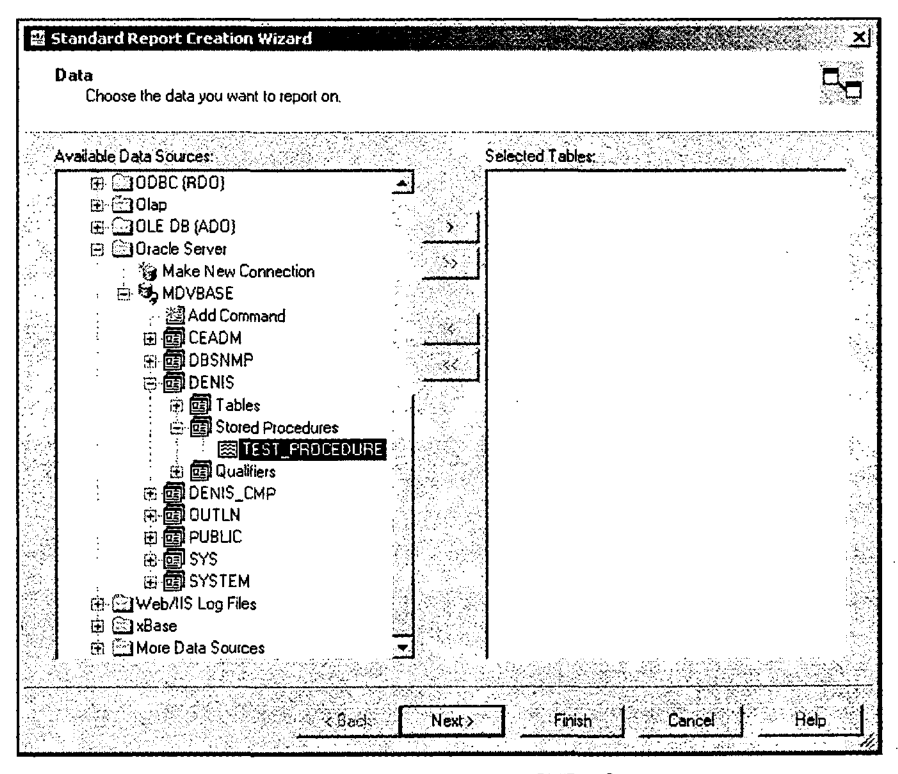


Рис. 7.30. Подключение к СУБД Oracle

Выбираем схему, в которой были созданы описанные выше элементы. В разделе **Stored Procedures** (Хранимые процедуры) выбираем хранимую процедуру TEST_PROCEDURE и добавляем ее в отчет в качестве источника данных. Так как процедура имеет параметры, автоматически появляется диалоговое окно **Enter Parameter Values** (Введите значения параметров), в котором можно установить требуемые значения так, как показано на рис. 7.31. Эти параметры автоматически будут включены в отчет и затем их можно будет менять.

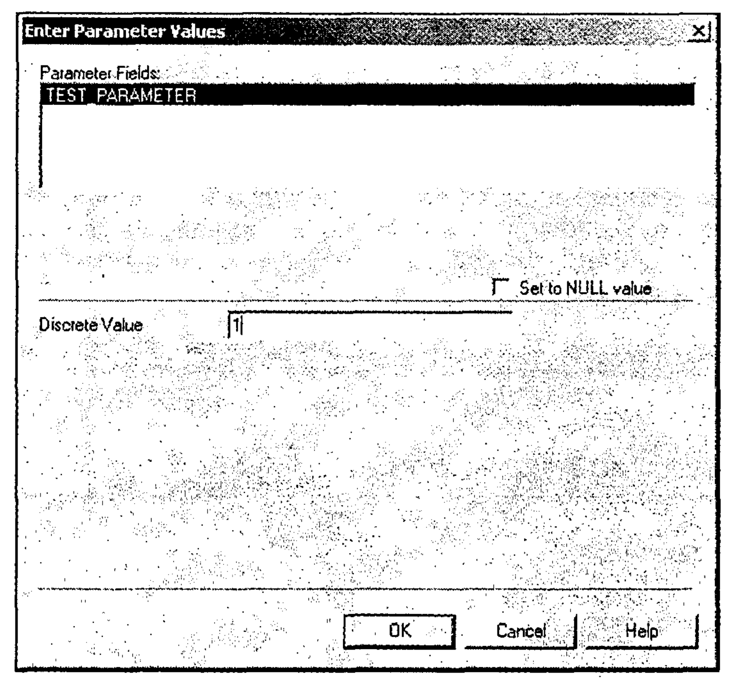


Рис. 7.31. Установка значений параметров для хранимой процедуры

Дальнейшая работа с полученными данными выполняется так же, как с таблицами и представлениями. То есть можно использовать формулы, функции, группировку и т. п.

Та же самая хранимая процедура может быть реализована с помощью пакета.

Создание пакета

```
CREATE OR REPLACE PACKAGE Test_Package
AS TYPE Test_Type IS REF CURSOR RETURN Test_Table%ROWTYPE;
PROCEDURE Test_Procedure (
Test_Cursor IN OUT Test_Type,
```

```
Test_Parameter IN Test_Table.ID%TYPE
);
END Test_Package;
/
```

Создание тела пакета

```
CREATE OR REPLACE PACKAGE BODY Test_Package
AS
PROCEDURE Test_Procedure (
Test_Cursor IN OUT Test_Type,
Test_Parameter IN Test_Table.ID%TYPE
) IS
BEGIN
OPEN Test_Cursor FOR
SELECT * FROM Test_Table
WHERE Test_Table.ID = Test_Parameter;
END Test_Procedure;
END Test_Package;
/
```

Единственное ограничение, которое накладывается на использование хранимых процедур, это запрет на организацию связи между процедурой и таблицами или представлениями, т. е. в одном отчете нельзя одновременно использовать хранимую процедуру и таблицу.

Упражнение 7.12

Создайте процедуру, которая возвращает цифровые данные в строковом формате прописью, и включите ее в отчет.

7.7. Изменение местоположения полей, проверка базы данных и смена драйвера

Иногда возникает необходимость во внесении изменений именно в ту базу, для которой вы создаете отчет. Встречаются случаи, когда база данных находится в состоянии непрерывного изменения, т. е. динамически меняется по мере создания отчета. Или, возможно, создается отчет для тестовой базы, а затем выполняется его связывание реальной базой данных. Желательно иметь возможность контролировать подобные изменения по мере обновления базы данных, без необходимости повторного создания каких-либо свойств или функций в отчете.

Одним из видов таких изменений базы данных может быть переименование полей разработчиком или администратором базы данных. Crystal Reports 9, обнаружив эти изменения, дает возможность изменить ссылки на поля в отчете, таким образом, новые имена полей можно автоматически связать с созданными ранее. Такая обработка полей освобождает разработчика от необходимости добавлять к отчету новые поля или модифицировать формулы, ссылающиеся на измененные поля.

Есть несколько способов согласовать изменения, сделанные в базе данных:

- проверить базу данных;
- указать местонахождение базы данных.

Все они описаны в этой главе.

7.7.1. Проверка или изменение местоположения базы данных

При внесении изменений в структуру базы данных, когда изменяются имена полей, типы данных и т. п., при перемещении базы данных в другое место или при необходимости привязки своего отчета к другой базе данных, у разработчика появится необходимость в использовании функций Crystal Reports 9, распознающих эти изменения.

В случае, когда разработчик базы данных добавил новые поля, удалил старые, изменил имена или типы данных существующих полей, при открытии отчета, основанного на этой информации, внесенные изменения не будут распознаны в автоматическом режиме. Даже после обновления отчета Crystal Reports 9 не обнаружит этих изменений. Единственный выход в данной ситуации — это сделать проверку соответствующей базы данных. Для этого необходимо выполнить команду меню **Database | Verify Database**. Если база данных не была изменена, то появится сообщение, показанное на рис. 7.32.

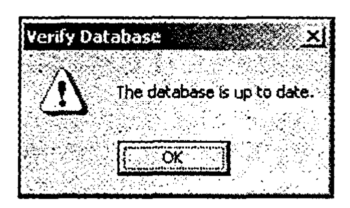


Рис. 7.32. Проверка базы данных. Изменений в базе данных не обнаружено

В противном случае, если база данных была изменена, появится сообщение подобное тому, которое изображено на рис. 7.33.

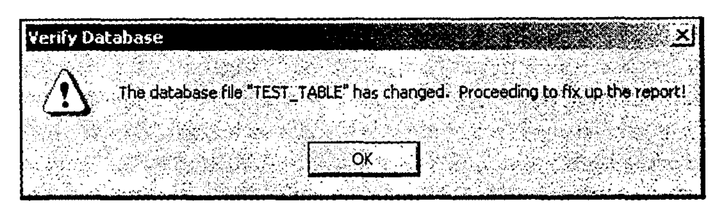


Рис. 7.33. Сообщение об изменении базы данных

При нажатии на кнопку **OK** Crystal Reports 9 вновь прочитает структуру базы данных и внесет необходимые изменения в имена таблиц, имена полей и типы данных. При запуске **Field Explorer** (Просмотрщик полей), можно будет увидеть произведенные изменения. Кроме того, если в отчете использовались связанные между собой таблицы и структура этих таблиц значительно изменилась, то, скорее всего, придется заново связать эти таблицы при помощи окна **Database Expert** (Эксперт базы данных). Если изменились имена полей, то появится диалоговое окно **Map Fields** (Карта соответствия полей), о котором будет рассказано далее.

Set Datasource Location
Change the location of the data source by selecting the current database (or table) and choosing the database (or table) to replace it with. Then click Update.
Current Data Source:
report
MDVBASE
Properties
ЗАКАЗЫ
ПОКУПАТЕЛИ
ПРОДАВЦЫ
ТОВАРЫ
Replace with:
Current Connections
Repository
Favorites
History
Create New Connection
Update
Close
Help

Рис. 7.34. Окно, позволяющее изменить местоположения источника

Иногда приходится менять скорее тип, а не имя базы данных. К примеру, часто бывает, что отчет, изначально разработанный для настольной базы данных, впоследствии используется для сетевого источника. В этом случае необходимо указать новое местонахождение базы данных и, возможно, изменить тип библиотек, используемых для соединения с источником. Для этого требуется вызвать с помощью команды меню **Database | Set Datasource Location** окно **Set Datasource Location**, показанное на рис. 7.34.

Как видно из этого рисунка, имеется возможность быстро изменить не только путь к источнику данных, но и его тип.

7.7.2. Согласование старых полей с новыми именами

Изменения в базе данных могут привести к несоответствиям с именами полей созданного ранее отчета. Например, отчет мог быть разработан для базы данных Microsoft Access, содержащей поле **Account Number** (пробел между словами важен для Crystal Reports), а реальности может понадобиться этот же отчет, но для работы с базой данных MS SQL Server, в которой это поле называется **Account_Number**, здесь пробел заменен знаком подчеркивания.

Что делать в случае изменения имен таблиц

Если разработчик или администратор базы данных изменит имя таблицы, при очередной попытке обновить отчет поступит сообщение об ошибке, указывающее, что данная таблица не может быть найдена. К сожалению, обновление отчета или проверка базы данных не решит эту проблему: сообщение будет повторяться, а отчет нельзя будет правильно сформировать.

Для решения этой проблемы следует выполнить следующие простые шаги:

1. Выполните команду меню **Database | Set Datasource Location**. В открывшемся диалоговом окне **Set Datasource Location** (Установка местонахождения источника данных), показанном на рис. 7.34, в разделе **Current Data Source** (Текущий источник данных), выберите таблицу, имя которой было изменено.
2. В окне **Replace with** укажите новый источник данных для выбранной таблицы.

Что делать в случае изменения полей таблиц

После выполнения перечисленных действий появится окно **Map Fields** (Карта соответствия полей), которое показано на рис. 7.35.

Список полей базы данных, предлагаемых для сопоставления полю отчета, меняется в зависимости от того, установлен ли флажок **Match Type** (Совпадение типа) в правой части диалогового окна. Если флажок установ-

лен, то будут показаны только поля того же типа данных (строка, число, дата и т. д.), что и выделенное поле отчета. Это помогает избежать неверного сопоставления, например, строкового поля в отчете числу или дате в новой базе данных, хотя иногда может потребоваться и такое сопоставление. Из-за несоответствия типов данных в правом списке может не оказаться поля, которое необходимо обработать. Например, при переходе от базы данных Microsoft Access к базе данных SQL Server, скорее всего не будут найдены поля, соответствующие полю валюты базы данных Access. В этом случае необходимо снять флажок **Match Type** (Совпадение тип), и найти поле для сопоставления.

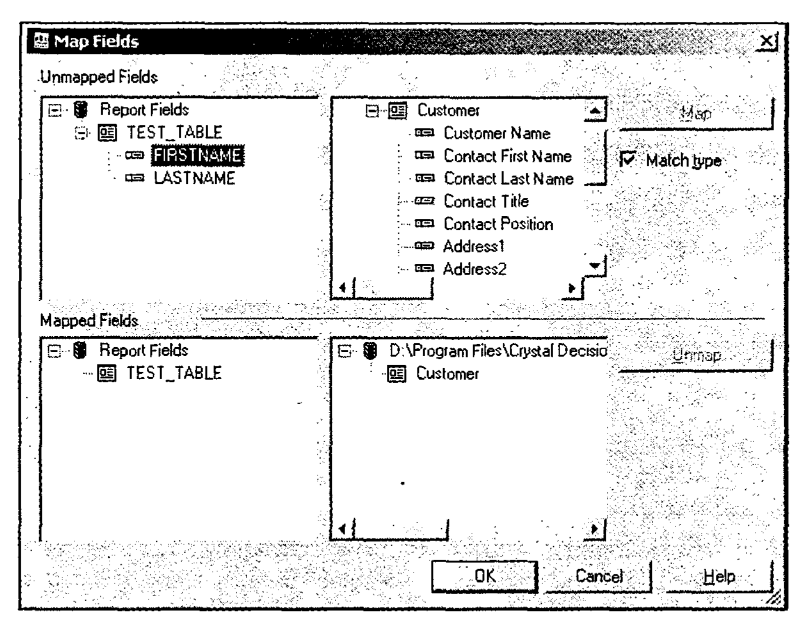


Рис. 7.35. Окно **Map Fields**, позволяющее сопоставить поля существующего отчета с новыми полями базы данных

При первом открытии диалогового окна некоторые поля могут уже присутствовать в двух нижних списках. Левый нижний список показывает поля отчета, которые уже имеют соответствующие поля в новой базе данных. В случае сопоставления полей из верхних списков, они также будут перемещены в нижние списки. Если существующее сопоставление предложено самим Crystal Reports 9 на основании совпадения имен полей или произошло ошибочное сопоставление полей, можно действовать так. Выделить поле в левом нижнем или в правом нижнем списке, при этом в другом списке будет выделено сопоставленное поле. Затем можно нажать кнопку **Unmap** (Отменить связь полей), чтобы отменить сопоставление полей и вернуть их в верхние списки.

Закончив сопоставление полей, необходимо нажать кнопку **OK**, диалоговое окно **Map Fields** (Карта соответствия полей) будет закрыто. Затем Crystal Reports 9 свяжет сопоставленные поля с новой базой данных. Если в других таблицах есть ненайденные поля, для каждой из них будет выведено диалоговое окно **Map Fields** (Карта соответствия полей).

Примечание

Все поля, которые не существуют в измененной базе данных, должны быть заменены новыми полями. Если вы не сопоставите новые имена полей старым, старые поля и объекты, на которых они основаны, будут удалены из вашего отчета!

7.8. Возможности по настройке отчетов для выполнения SQL-запросов, формируемых Crystal Reports на стороне сервера базы данных

В связи с тем, что в большинстве случаев выполнение функций на стороне сервера происходит значительно быстрее, чем на стороне клиента, Crystal Reports 9 имеет ряд возможностей по настройке для обработки запросов на сервере.

Для того чтобы выполнять группирование на сервере, а не группировать данные в Crystal Reports 9 после их выборки, необходимо, чтобы отчет удовлетворял следующим требованиям:

- должна быть выбрана опция **Perform Grouping On Server** (Выполнять группирование на сервере), которая задается через команду меню **Database | Perform Grouping On Server**;
- отчет должен содержать условие группирования;
- желательно, чтобы все имеющиеся секции **Details** были скрыты, хотя в ряде случаев они скрываются автоматически;
- сложные формулы, вычисляемые в секции **Details**, недопустимы. Необходимо использовать SQL-выражения;
- не должно быть вычисления кумулятивных значений;
- агрегатные функции типа Average и Distinct использовать нельзя. Возможны только варианты функций, которые имеют аналоги в SQL. Это SUM, COUNT и т. п.;
- параметры, задаваемые через **Group Sort Expert** (Эксперт группировки и сортировки данных), и специальные типы группировки не должны быть задействованы.

Если отчет удовлетворяет перечисленным условиям, в SQL-запрос добавляется предложение GROUP BY, которое позволяет выполнить группировку на сервере.



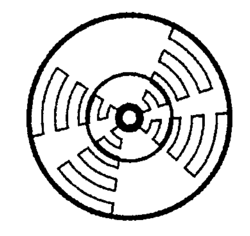
Как видно из перечисленного списка требований, для того, чтобы отчет обрабатывал запрос на сервере, а не на клиентском месте, необходимо сильно ограничить использование функциональности Crystal Reports 9. Однако в ряде случаев, такие возможности могут значительно ускорить обработку данных.

В случае если выбрана опция **Use Indexes Or Server For Speed** (Использовать индексы или сервер для ускорения обработки данных) на вкладке **Database** диалогового окна **Options**, которое вызывается с помощью команды меню **File | Options**, в SQL-запрос автоматически добавляется предложение ORDER BY. Кроме того, иногда для настольных источников данных будет формироваться SQL-запрос, оптимизированный под использование механизмов обработки данных на выбранном сервере.

Упражнение 7.13

Включите опции Crystal Reports 9, относящиеся к настройке механизмов работы с базами данных, и модифицируйте конструкцию отчета. Просмотрите результирующий запрос, выполнив команду меню **Database | Show SQL Query**.

Глава 8



Распространение отчетов

8.1. Экспорт отчетов

Часто возникает потребность получить готовый отчет в формате, который не потребует установки Crystal Reports 9 или его компонентов на машине пользователя. При этом желательно, чтобы элементы форматирования сохранялись как можно точнее. Crystal Reports 9 позволяет выполнять экспорт отчетов в любом из перечисленных форматов:

- Character-separated values
- Comma-separated values (CSV)
- Crystal Reports (RPT)
- Microsoft® Excel
- HTML 4.0
- HTML 3.2
- ODBC
- Adobe Acrobat (PDF)
- Record style
- Report Definition
- Rich Text Format
- Tab-separated text
- Tab-separated values
- Text
- Word for Windows document
- XML

Для того чтобы приступить к выполнению экспорта, необходимо нажать кнопку или выполнить команду меню **File | Export**. Появится диалоговое окно **Export**, показанное на рис. 8.1.

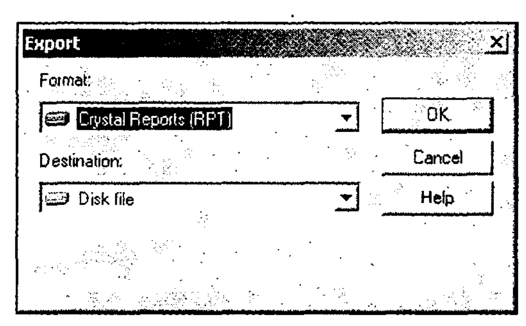


Рис. 8.1. Диалоговое окно настройки параметров экспорта

В этом окне можно выбрать параметры экспорта отчета. Из выпадающего списка **Format** выбирается требуемый формат. Кроме этого, имеется возможность выбора механизма сохранения или обработки экспортированного отчета. Список возможных вариантов обработки представлен в поле **Destination**:

- **Application** — можно сразу открыть результат экспорта с помощью приложения. Например, при экспорте в Word или RTF автоматически запустится Word for Windows и покажет отчет;
- **Disk file** — сохранение в виде файла, который потом можно обрабатывать;
- **Exchange Folder** — сохранение в папку Microsoft Exchange;
- **Lotus Domino** — сохранение в базе данных Lotus Domino;
- **Lotus Domino Mail** — отправка электронной почты посредством Lotus Domino;
- **Microsoft Mail (MAPI)** — отправка электронной почты посредством механизма фирмы Microsoft.

После того как выбран формат экспорта и указан механизм его обработки, можно нажать кнопку **OK**. Для большинства форматов непосредственно перед выполнением экспорта появится диалоговое окно **Export Options** (рис. 8.2). Используя настройки в этом окне, можно определить диапазон экспортируемых страниц:

- **All** — все страницы отчета;
- **Page Range From: To:** — только страницы, номера которых попадают в указанный в полях диапазон.

После того как определен диапазон экспортируемых страниц и нажата кнопка **OK**, появляется стандартное окно, в котором необходимо указать каталог и имя создаваемого файла, если выбрана опция **Disk File**. Для других опций надо будет указать адресата, которому отправляется отчет по почте, либо каталог Microsoft Exchange. После выполнения этих действий открывается

диалоговое окно **Exporting Records**, которое показывает прогресс экспорта данных. Внешний вид окна показан на рис. 8.3.

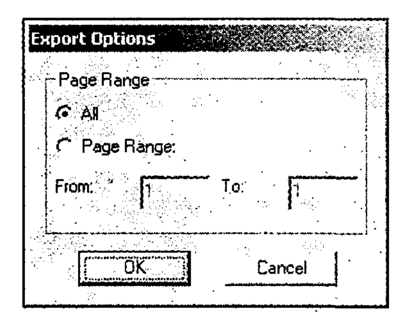


Рис. 8.2. Окно **Export Options**

Рис. 8.3. Окно **Exporting Records**

В случае экспорта отчета в формат MS Excel 97—MS Excel 2000 должно появится диалоговое окно **Excel Format Options**, которое показано на рис. 8.4.

Рис. 8.4. Окно **Excel Format Options**. Настройка параметров экспорта

Параметры настройки в данном окне можно разделить на группы:

- **Column width** — настройка параметров ширины колонок;
- **Page range** — выбор диапазона экспортируемых страниц отчета;
- дополнительные параметры.

Параметры, относящиеся к ширине экспортируемых колонок, содержат следующие варианты настроек:

- **Column width based on objects in the** — ширина колонок определяется шириной объекта из выпадающего списка:
  - Whole Report — ширина колонок определяется шириной объектов всего отчета;
  - Report Header — ширина колонок определяется шириной объектов, находящихся в секции Report Header;
  - Group Header #N — ширина колонок определяется шириной объектов, находящихся в секции Group Header. N заменяется на номер секции;
  - Group Footer #N — ширина колонок определяется шириной объектов, находящихся в секции Group Footer. N заменяется на номер секции;
  - Page Footer — ширина колонок определяется шириной объектов, находящихся в секции Page Footer.
- **Constant column width (in points)** — одинаковая ширина для всех колонок, ширину колонки необходимо вписать в поле.

Секция **Page range** содержит те же настройки, что присутствуют в диалоговом окне **Export Options**, показанном на рис. 8.2.

Дополнительные параметры экспорта позволяют выполнить следующие действия:

- **Export Page Header and Page Footer** — выполнять экспорт данных, которые содержатся в секциях отчета Page Header и Page Footer;
- **Create page breaks for each page** — формировать разрыв страницы в итоговом документе;
- **Convert date values to string** — выполнять конвертацию полей типа Date в строку.

После выполнения настроек в окне **Excel Format Options** потребуется выполнить те же действия, что описаны выше для экспорта в другие форматы.

Помимо экспорта данных из отчета в файл выбранного формата существует возможность экспорта с помощью ODBC. Данный механизм предполагает создание в выбранном ODBC-источнике новой таблицы либо использование уже существующей таблицы для сохранения информации из отчета. При экспорте данных с использованием ODBC диалоговое окно **Export** будет выглядеть так, как это показано на рис. 8.5.

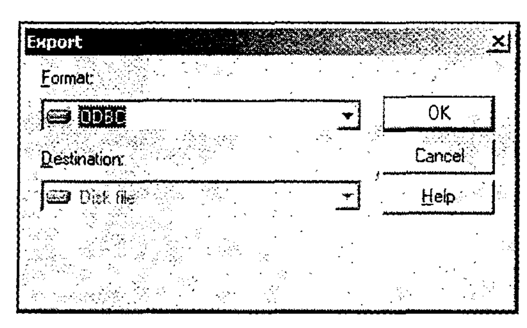


Рис. 8.5. Диалоговое окно **Export**. Использование ODBC

На следующем шаге, при выполнении экспорта с помощью ODBC, открывается окно **ODBC Formats**, содержащее список всех доступных драйверов.

ODBC Formats
Crystal Web Logs
DeluxeCD
ECDCMusic
Export Report
FoodMart
FoodMart 2000
MQIS
ORAODBC
PBServer
scalaDB_ODBC
Schearing_Data
Schering_AG_MDB
Schering_AG_SQL
SQLBase
OK
Cancel

Рис. 8.6. Окно **ODBC Formats**

После выбора готового источника или создания нового необходимо вписать имя таблицы, в которую будут записаны данные из отчета, в диалоговом окне **Enter ODBC Table Name**, показанном на рис. 8.7.

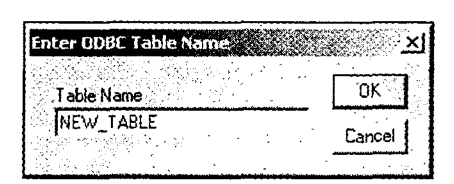


Рис. 8.7. Окно **Enter ODBC Table Name**

Далее нажатием кнопки **OK** выполняется экспорт данных в указанную таблицу. В случае если указанной таблицы не существует, она будет создана, и затем в нее запишутся данные из отчета. Если будет указано имя таблицы, которая уже есть в выбранном источнике, появится предупреждение,

показанное на рис. 8.8. Данное предупреждение предлагает выбрать другое имя для таблицы либо предварительно удалить существующую таблицу.

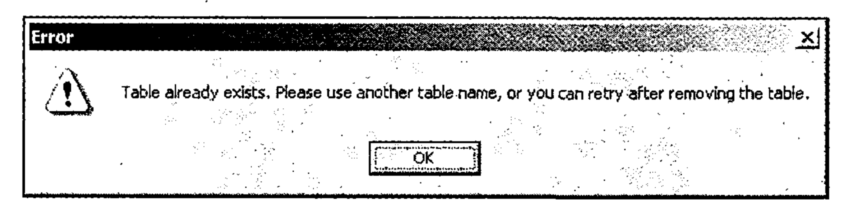


Рис. 8.8. Предупреждение об экспорте в существующую таблицу



Для экспорта данных в ODBC-источник можно использовать только табличную форму отчетов. Все другие варианты отчетов будут сохранены некорректно либо с ошибками.

Упражнение 8.1

1. Экспортируйте отчет с подотчетами в MS Word или MS Excel.
2. Экспортируйте отчет через ODBC в любую базу данных и посмотрите результат.

8.2. Распространение отчетов с помощью Crystal Enterprise или Crystal Report Application Server

В большинстве современных компаний стоит задача не только сформировать огромный набор отчетов, но и быстро доставить эти отчеты до конечного пользователя. Одним из вариантов решения такой задачи может считаться установка Crystal Reports на рабочие места пользователей. Однако это повлечет за собой необходимость в обучении пользователей работе с Crystal Reports. Кроме того, часто рабочие станции у конечных пользователей имеют слишком ограниченные возможности для установки Crystal Reports, или в соответствии с политикой фирмы устанавливать специализированное программное обеспечение можно только со специального разрешения системного администратора и руководства. Для обеспечения быстрой и экономной доставки отчетов до конечного пользователя в Crystal Reports включены механизмы, позволяющие работать с отчетами, используя Web-технологии и не устанавливая дополнительного программного обеспечения.

В различных версиях Crystal Reports имеются разные компоненты, обеспечивающие доступ к отчетам из Интернет. В Crystal Reports 8.0 есть два ком-

понента, которые позволяют вызвать отчеты из браузера, такого как Internet Explorer или Netscape Communicator:

- Crystal Web Component Server
- Crystal Page Server

Подробно механизмы работы с данными компонентами описаны в книге "Введение в Crystal Reports", Маклаков С., Матвеев Д. Интерфейс-Пресс. Богородский печатник.

Начиная с Crystal Reports версии 8.5 и выше, вместо этих компонентов появилась самостоятельная специализированная среда распространения и администрирования отчетов Crystal Enterprise. Crystal Enterprise в отличие от Crystal Web Components предоставляет следующие возможности:

- средства быстрого распространения отчетов;
- средства администрирования;
- средства обработки отчетов по расписанию;
- механизмы локализации.

Так же помимо Crystal Enterprise есть возможность распространения отчетов с помощью Crystal Report Application Server (RAS). Внешний вид и основная функциональность Crystal Report Application Server такие же, как и у Crystal Enterprise.

Примечание

Пробная версия среды распространения отчетов Crystal Report Application Server поставляется совместно с Crystal Reports 9.

8.2.1. Архитектура Crystal Enterprise 9

Система Crystal Enterprise является многозвенной. Все компоненты этой системы предназначены для выполнения различных задач. Причем некоторые из компонентов можно устанавливать не только на компьютеры, находящиеся под управлением MS Windows 9X, ME, NT или 2000, но и на машины, управляемые UNIX или Linux. Crystal Enterprise имеет четыре основных звена (схема 8.1):

- клиентское звено
- интеллектуальное звено
- исполнительное звено
- звено данных

Из схемы видно, что каждое звено Crystal Enterprise состоит из нескольких компонентов.

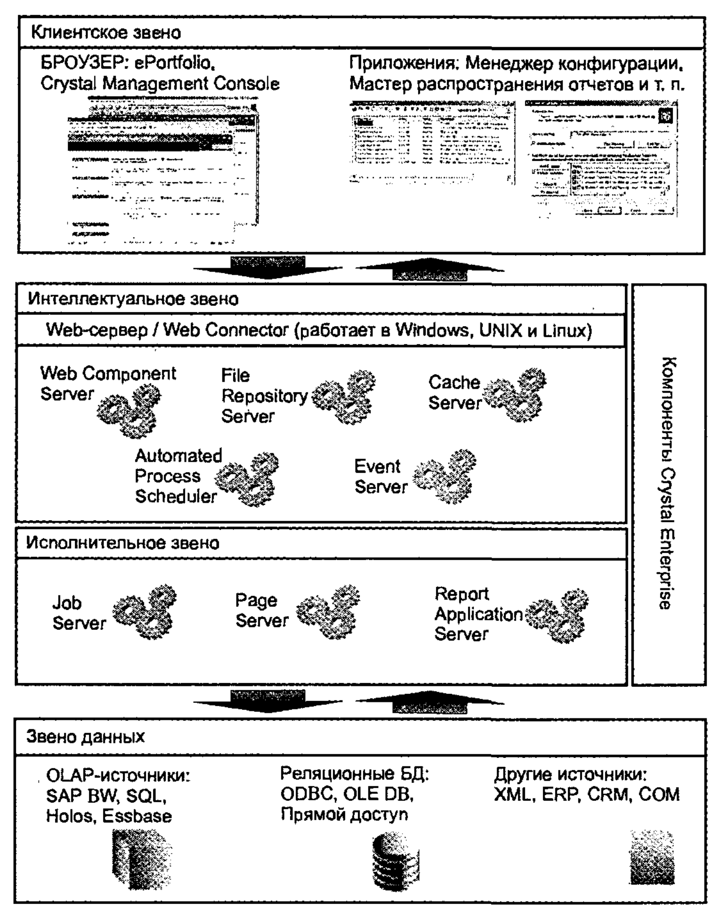


Схема 8.1. Структура звеньев Crystal Enterprise

Состав клиентского звена:

- элементы работы через браузер (Internet Explorer или Netscape Communicator):
  - **ePortfolio** — рабочее окно конечного пользователя, которое вызывается из браузера и не требует установки дополнительного программного обеспечения на компьютер пользователя;
  - **Crystal Management Console** — рабочее место администратора Crystal Enterprise, которое вызывается удаленно из браузера и не требует установки дополнительного программного обеспечения на компьютер администратора;
- приложения:
  - Crystal Configuration Manager — утилита администратора для выполнения настройки всех серверных компонентов Crystal Enterprise. Может

находиться как на машине с Crystal Enterprise, так и на машине администратора;

- Crystal Publishing Wizard — утилита разработчика, позволяющая быстро разместить необходимые для работы отчеты на сервере Crystal Enterprise, выполнив предварительную настройку параметров;
- Crystal Import Wizard — утилита администратора, позволяющая перенести информацию о пользователях, привилегиях и отчетах из Seagate Info 7.X и Crystal Enterprise более ранних версий;
- Crystal Web Wizard — утилита, позволяющая разработчику настроить новый вариант Crystal Enterprise.

Интеллектуальное звено содержит следующие компоненты:

- Web Component Server — это мост между Web Connector и остальными компонентами Crystal Enterprise. Данный компонент Crystal Enterprise выполняет обработку всех запросов от клиентов и возвращает результаты, выступая в роли сервера приложений.
- Web Connectors — компоненты, которые организуют связь Web Component Server с Web-сервером. В зависимости от используемого Web-сервера и операционной системы можно подключать различные Web Connector.
- Automated Process Scheduler — служба, обеспечивающая функции управления базой данных Crystal Enterprise:
  - обеспечивает защиту информации от несанкционированного доступа;
  - управляет объектами в Crystal Enterprise;
  - управляет всеми остальными компонентами Crystal Enterprise.
- File Repository Server — состоит из двух компонентов:
  - Input File Repository Server — управляет всеми отчетами, которые были опубликованы администратором или пользователем;
  - Output File Repository Server — управляет всеми экземплярами отчетов, которые были созданы Job Server.
- Event Server — управляет событиями, связанными с файлами. При использовании некоторых настроек Crystal Enterprise можно отслеживать изменения, которые происходят в определенных каталогах с указанными файлами, и выполнять действия с отчетами, опубликованными в Crystal Enterprise.
- Cache Server — управляет процессами кэширования информации, запрошенной клиентом через Web Component Server.

Исполнительное звено включает в себя следующий набор компонентов:

- Job Server — выполняет обработку отчетов по расписанию, на основании данных, переданных из Automated Process Scheduler, и формирует требуемые экземпляры отчетов, содержащие запрошенные данные;

- Page Server — выполняет обработку запросов к отчетам и отвечает за генерацию страниц в формате Encapsulated Page Format. Страницы данного формата содержат информацию о том, как необходимо отображать отчет у клиента. Page Server получает данные непосредственно из базы данных или из последнего экземпляра отчета;
- Report Application Server — выполняет функции, сходные с Page Server. Данный компонент, как уже было сказано ранее, входит в комплект поставки Crystal Reports.

Звено данных организует доступ к множеству различных источников данных. Crystal Reports и Crystal Enterprise позволяют использовать большинство современных баз данных в качестве источника данных для отчетов.

8.2.2. Основы администрирования Crystal Enterprise

Для того чтобы организовать корректную доставку отчетов конечному пользователю и обеспечить безопасность работы в сети, необходимо наличие средств администрирования и публикации. В Crystal Enterprise есть большой набор компонентов для администрирования. К таким компонентам относятся следующие приложения:

- Crystal Configuration Manager;
- Crystal Management Console;
- Crystal Publishing Wizard;
- Crystal Web Wizard.

Crystal Configuration Manager является приложением, которое ставится на компьютер с установленными основными компонентами Crystal Enterprise или на компьютер администратора. С помощью данной утилиты администратор может выполнять различные задачи по управлению и настройке служб, входящих в состав Crystal Enterprise. Программа Crystal Configuration Manager вызывается с помощью команды меню **Start | Programs | Crystal Enterprise 9 | Crystal Configuration Manager**. При этом появляется окно, показанное на рис. 8.9.

В окне на рисунке изображен список серверных компонентов Crystal Enterprise, т. е. компонентов, без которых Crystal Enterprise работать не будет. Данное окно полностью совпадает по своим свойствам и возможностям с окном Microsoft Management Console (MMC). Используя приложение Crystal Management Console, администратор может выполнить следующие действия:

- завершить работу или запустить любой компонент из списка;
- изменить параметры работы у любого из компонентов, представленных в списке;

- переопределить источник данных, используемый для хранения информации о Crystal Enterprise.

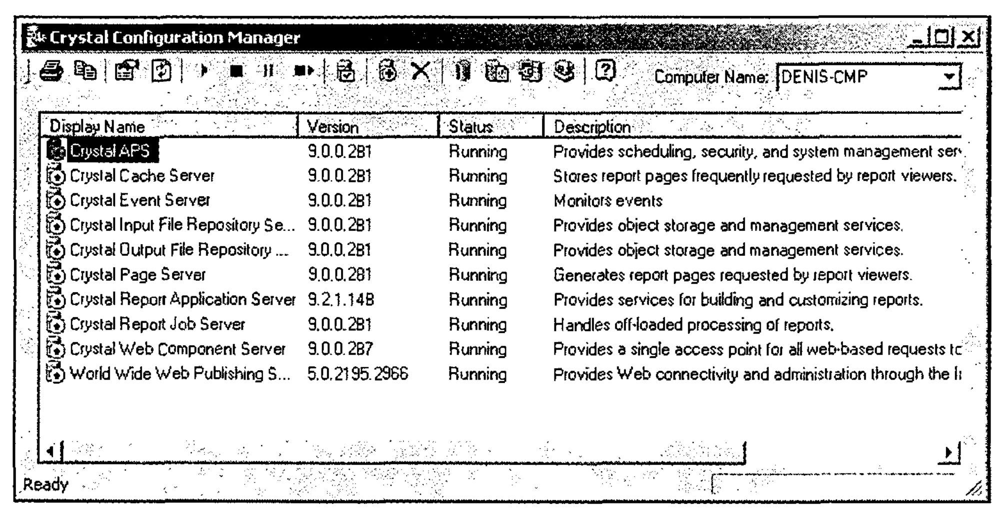


Рис. 8.9. Окно **Crystal Configuration Manager**

Основные действия, которые можно выполнять в окне Crystal Configuration Manager:

- завершить работу любого из компонентов Crystal Enterprise, для этого необходимо установить фокус на компонент в списке и нажать кнопку ■ либо выполнить команду контекстного меню **Stop**;
- запустить неработающий компонент Crystal Enterprise, для этого надо установить фокус на требуемый компонент в списке и нажать кнопку ▶ либо выполнить команду контекстного меню **Start**;
- перезапустить компонент Crystal Enterprise или компонент, представленный в списке, для этого необходимо установить фокус на компонент в списке и нажать кнопку ■▶ либо выполнить команду контекстного меню **Restart**.

Большинство компонентов, представленных в окне Crystal Configuration Manager, имеют идентичный набор параметров. Для их просмотра и изменения необходимо завершить работу интересующего компонента и, нажав кнопку либо вызвав контекстное меню, выполнить команду **Properties**. При этом появится диалоговое окно свойств выбранного компонента. Один из возможных вариантов показан на рис. 8.10.

В ряде случаев возможны некоторые отличия. Однако дополнительные параметры в диалоговом окне свойств или являются нередактируемыми, или же их настройку лучше выполнять с помощью Crystal Management Console.

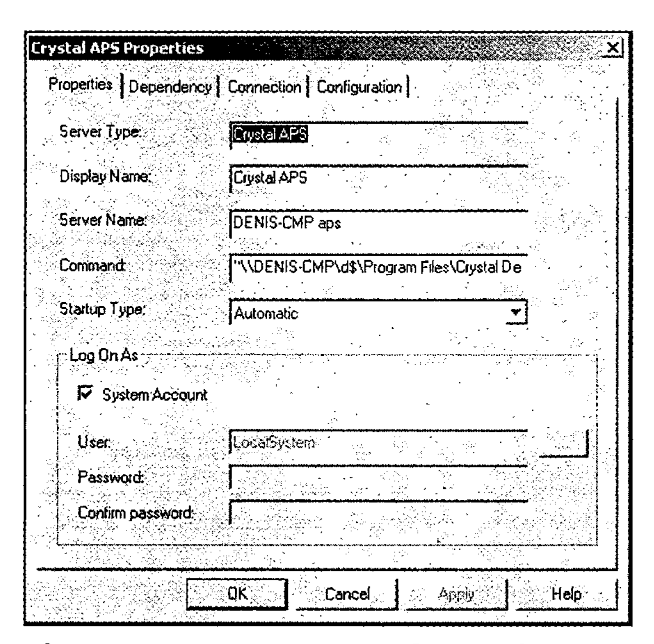


Рис. 8.10. Окно настройки параметров для Crystal APS. Аналогичные окна отображаются для других компонентов Crystal Enterprise

Независимо от того, настройка какого компонента выполняется, диалоговое окно свойств содержит следующие вкладки:

- **Properties** — основные свойства выбранного компонента Crystal Enterprise;
- **Dependency** — список приложений, от которых зависит корректная работа выбранного компонента Crystal Enterprise;
- **Connection** — список IP-адресов, которым разрешен доступ к компонентам и отчетам Crystal Enterprise;
- **Configuration** — параметры конфигурации, для некоторых компонентов они могут иметь ряд отличительных особенностей.

На вкладке **Properties** могут быть заданы следующие параметры:

- Server Type — тип сервера;
- Display Name — имя, отображаемое при просмотре списка компонентов;
- Server Name — физическое имя сервера;
- Command — команда, с помощью которой выполняется запуск сервера;

- Startup Type — тип запуска, принимающий значения из списка:
  - Automatic — запуск сервера производится в момент загрузки операционной системы;
  - Disabled — сервер недоступен;
  - Manual — сервер запускается вручную;
- Logon As с возможными значениями:
  - System Account — в случае, когда данный параметр активен, загрузка сервера производится независимо от того, кто запустил компьютер, иначе используются параметры, описанные ниже;
  - User — имя пользователя в домене или на локальном компьютере;
  - Password — пароль пользователя;
  - Confirm Password — подтверждение пароля пользователя.

Кроме описанных вкладок в диалоговом окне настройки свойств, для Crystal Report Application Server и Crystal Report Job Server, имеется еще вкладка **Parameters**, которая позволяет определить некоторые специфические параметры для указанных служб.

Crystal Enterprise допускает хранение информации о пользователях, привилегиях и отчетах в различных типах источников данных. При установке Crystal Enterprise обычно в качестве хранилища используется MSDE или MS SQL Server, если его компоненты установлены заранее. Однако во многих случаях по различным соображениям использование MSDE или MS SQL Server недопустимо. Для таких случаев предусмотрена возможность переноса хранилища в другой вариант источника данных. Допустимые источники данных показаны в диалоговом окне **Select Database Driver** на рис. 8.11.

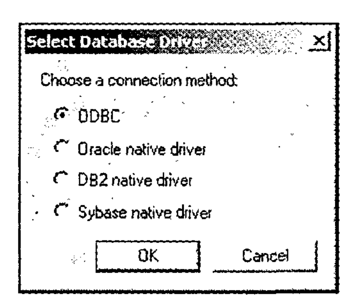


Рис. 8.11. Диалоговое окно **Select Database Driver**. Выбор источника данных для хранилища Crystal Enterprise

Для того чтобы сменить источник данных, используемый в качестве хранилища, и перенести имеющуюся информацию, необходимо выполнить следующие шаги:

1. Завершить работу компонента Crystal APS *(см. разд. 8.2.2)*.

Crystal APS Properties
Properties | Dependency | Connection | Configuration
Command Line Options
Port Number:
6400
Use default port number.
APS Data Source
CE90
Specify...
APS belongs to cluster "DENIS-CMP".
Change Cluster Name to
OK
Cancel
Apply
Help

Рис. 8.12. Диалоговое окно **Crystal APS Properties**. Вкладка **Configuration**

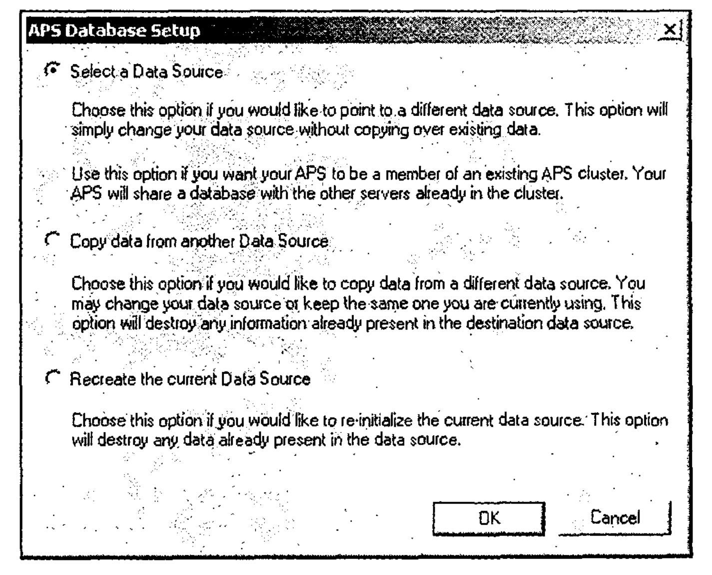


Рис. 8.13. Окно **APS Database Setup**

2. Вызвать диалоговое окно **Crystal APS Properties** и перейти на вкладку **Configuration** (рис. 8.12).
3. На вкладке **Configuration** нажать кнопку **Specify**, после чего откроется диалоговое окно **APS Database Setup**, см. рис. 8.13. Тот же результат будет получен в случае, если вместо вызова окна **Crystal APS Properties** нажать кнопку [icon].
4. В окне **APS Database Setup** необходимо выбрать один из вариантов конфигурирования источника данных и нажать кнопку **OK**. Список возможных действий выглядит так:
   - Select a Data Source — выбор нового источника данных для хранилища Crystal Enterprise.
   - Copy data from another Data Source — копирование информации в новое хранилище из использовавшегося ранее.
   - Recreate the current Data Source — создание пустого хранилища в текущем источнике.
5. В случае выбора опции Select a Data Source появится диалоговое окно, показанное на рис. 8.11. В этом окне необходимо выбрать желаемый вариант источника и, если потребуется, указать его параметры (рис. 8.14):
   - имя сервера базы данных;
   - имя пользователя;
   - пароль пользователя.

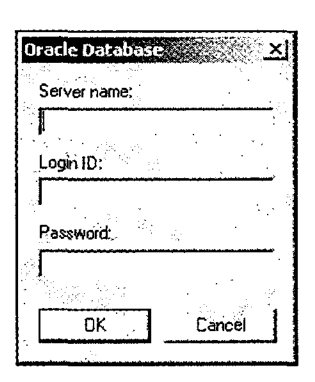


Рис. 8.14. Окно определения параметров подключения к выбранной базе данных

6. Если параметры определены корректно, то будет выполнено переключение на новый источник. Для того чтобы не возникало сообщений об ошибке, необходимо в окне **Crystal APS Properties** на вкладке **Configuration** (рис. 8.12) сделать активным параметр **Change Cluster Name to** и вписать в поле имя компьютера, на котором установлен Crystal Enterprise.

7. Когда производится подключение к новому источнику данных, нужно выполнить перенос информации о пользователях, привилегиях и отчетах из ранее использовавшегося хранилища. Для этого необходимо выполнить действия п. 2 и п. 3. В окне **APS Database Setup** выбрать параметр **Copy data from another Data Source** и нажать кнопку **OK**.
8. В диалоговом окне **Specify Data Source**, показанном на рис. 8.15 из выпадающего списка **Source contains data from version** выбрать один из вариантов:
   - Autodetect — автоматическое определение типа источника;
   - Crystal Enterprise 8.0;
   - Crystal Enterprise 8.5;
   - Crystal Enterprise 9.0.

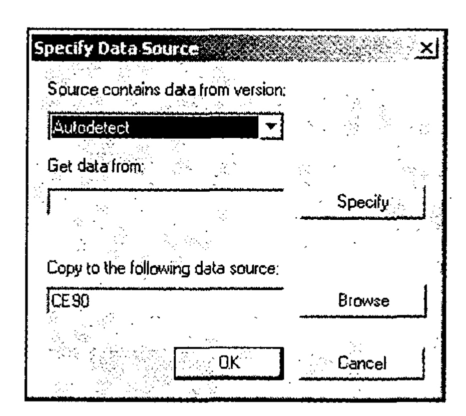


Рис. 8.15. Окно **Specify Data Source**. Выбор источников для синхронизации данных

9. Далее, нажав кнопку **Specify**, находящуюся напротив поля **Get Data from**, выбрать источник, из которого необходимо перенести данные. Механизм выбора совпадает последовательностью действий, описанной в п. 5.
10. В поле **Copy to the following data source** также можно указать источник, куда требуется перенести информацию.
11. При нажатии кнопки **OK** будет произведено копирование данных между выбранными источниками.

Если новый источник данных позволяет выполнять подключение к нему, используя учетную запись операционной системы, то можно, не изменяя ничего более, запустить Crystal APS. При корректном выполнении операций по переключению на новый источник данных Crystal APS запустится, после чего можно будет приступать к работе с Crystal Enterprise. Если новое хранилище Crystal Enterprise требует подключения через учетную запись зареги-

стрированного в хранилище пользователя, необходимо перед запуском Crystal APS выполнить три действия.

1. С помощью административных утилит операционной системы зарегистрировать пользователя, имя которого и пароль совпадают с именем и паролем пользователя, являющегося владельцем хранилища Crystal Enterprise.
2. Вызвать диалоговое окно **Crystal APS Properties** и на вкладке **Properties** в разделе **Log On As** сделать неактивным параметр **System Account**, а в поля **User**, **Password** и **Confirm Password** вписать имя созданного на первом шаге пользователя или выбрать его из списка, его пароль, подтверждение пароля.
3. Далее можно запустить Crystal APS, предварительно закрыв окно **Crystal APS Properties**.

Для удаленного администрирования компонентов Crystal Enterprise можно воспользоваться приложением, которое использует Web-технологии и не требует установки дополнительных программ. Данное приложение требует наличия на компьютере только браузера (Internet Explorer или Netscape Communicator) и возможности выхода в сеть. Запуск административного элемента Crystal Enterprise осуществляется выполнением команды меню **Start | Programs | Crystal Enterprise 9 | Crystal Launchpad**. При выполнении этой команды появляется окно браузера с открытой стартовой страницей **Crystal Enterprise** так, как это показано на рис. 8.16.

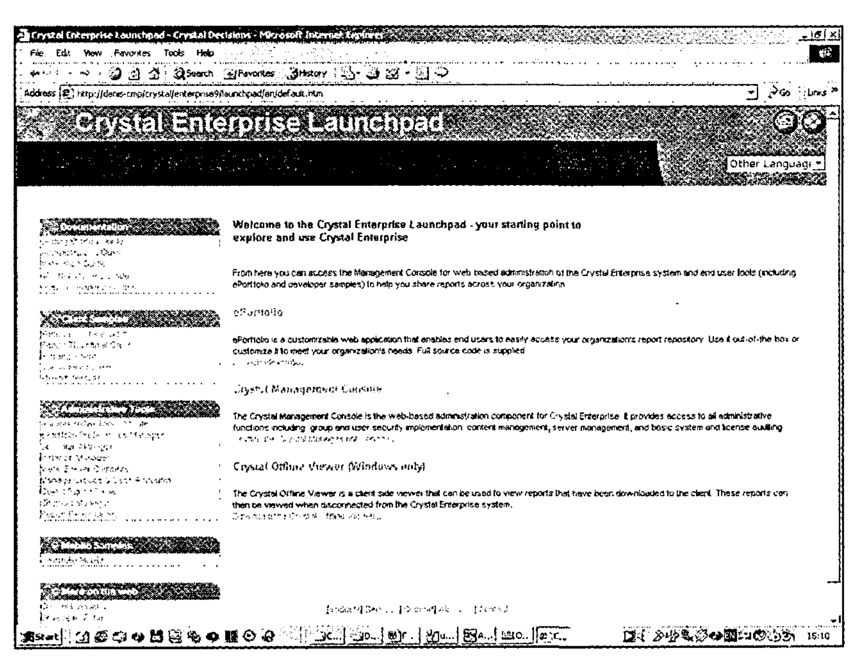


Рис. 8.16. Окно **Crystal Launchpad**

На этой странице необходимо выбрать ссылку Crystal Management Console. При выполнении перехода по этой ссылке откроется страница **Crystal Management Console Log On**, показанная на рис. 8.17.

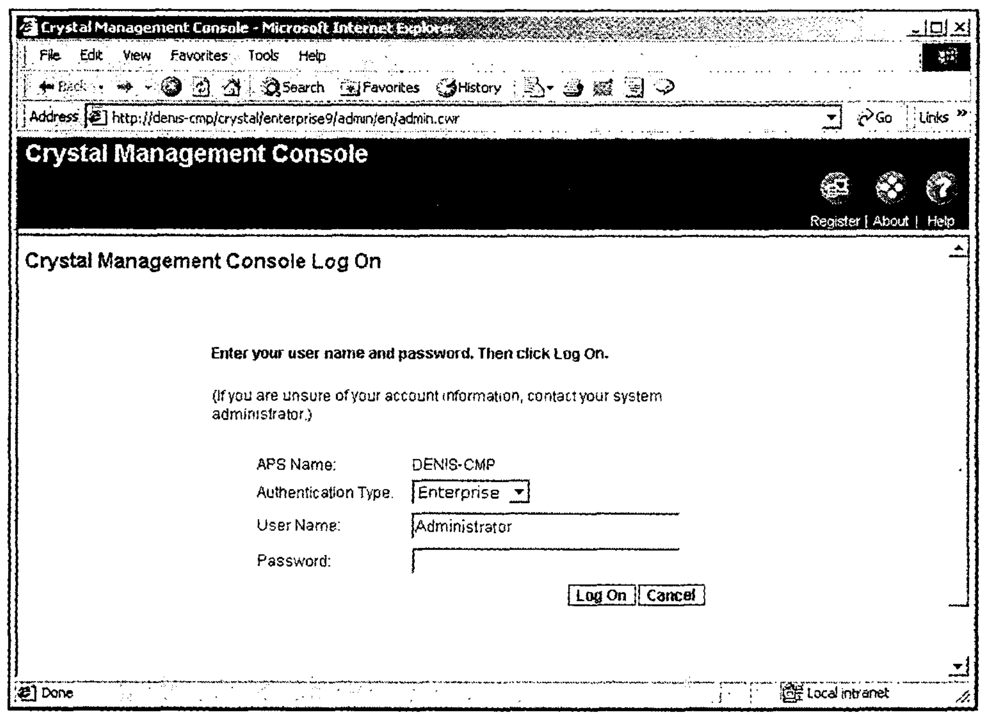


Рис. 8.17. Страница **Crystal Management Console Log On**

На этой странице необходимо выбрать параметры подключения к Crystal Enterprise.

- **Authentication Type** — способ авторизации:
  - LDAP (Lightweight Directory Access Protocol).
  - Enterprise — только для зарегистрированных пользователей Crystal Enterprise.
  - Windows NT — для пользователей, которые имеют право использовать учетные записи операционной системы.
- **User Name** — имя пользователя Crystal Enterprise, при установке Crystal Enterprise создается пользователь Administrator.
- **Password** — пароль пользователя Crystal Enterprise (пользователь Administrator по умолчанию не имеет пароля).

После ввода параметров подключения и нажатия кнопки **Log On** открывается рабочая страница **Crystal Management Console**, показанная на рис. 8.18.

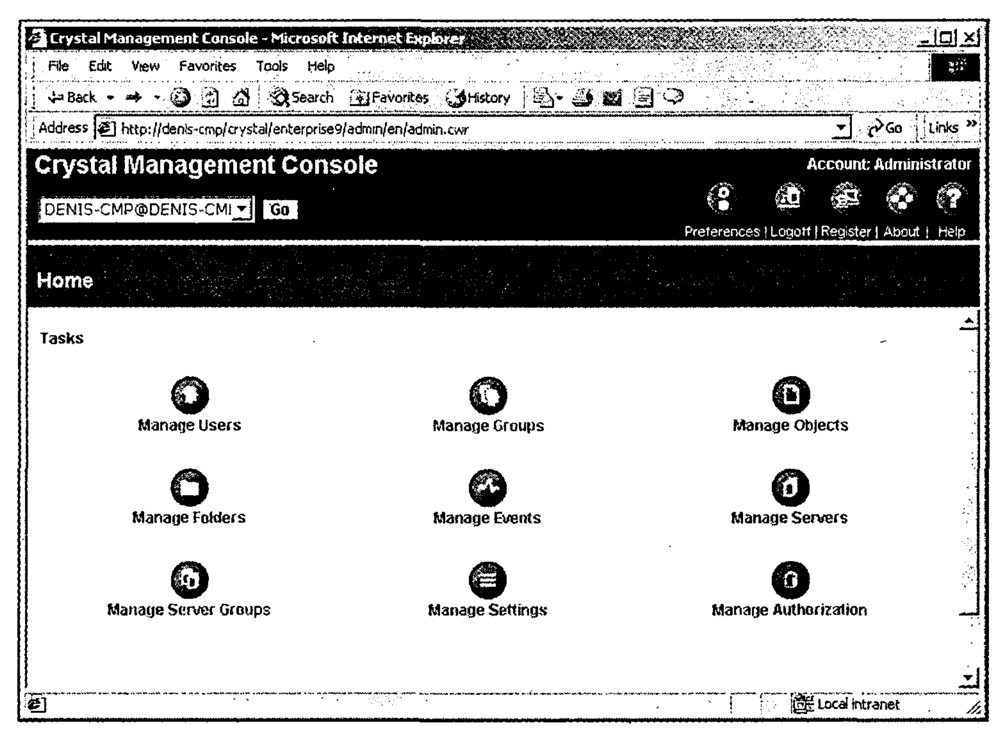


Рис. 8.18. Crystal Management Console. Основная рабочая страница

Ссылки на данной странице можно определить следующим образом:

- **Manage Users** — раздел, используемый для просмотра и администрирования пользователей;
- **Manage Groups** — раздел, используемый для просмотра и администрирования групп пользователей;
- **Manage Objects** — раздел, используемый для просмотра и публикации отчетов;
- **Manage Folders** — раздел, используемый для просмотра, редактирования и создания папок;
- **Manage Events** — раздел, используемый для настройки событий, которые относятся к обработке отчетов;
- **Manage Servers** — раздел, используемый для администрирования компонентов Crystal Enterprise по аналогии с Crystal Configuration Manager;

- **Manage Server Groups** — раздел, используемый для просмотра и администрирования групп серверов Crystal Enterprise, которые могут работать под управлением различных операционных систем в единой сети;
- **Manage Settings** — раздел, позволяющий настроить основные параметры работы Crystal Enterprise;
- **Manage Authorization** — раздел, содержащий информацию о лицензии и доступных способах авторизации.

Перейдя по ссылке **Manage Users**, администратор имеет возможность:

- просмотреть список уже зарегистрированных пользователей;
- зарегистрировать нового пользователя (если разрешает лицензия);
- изменить список привилегий, если это необходимо.

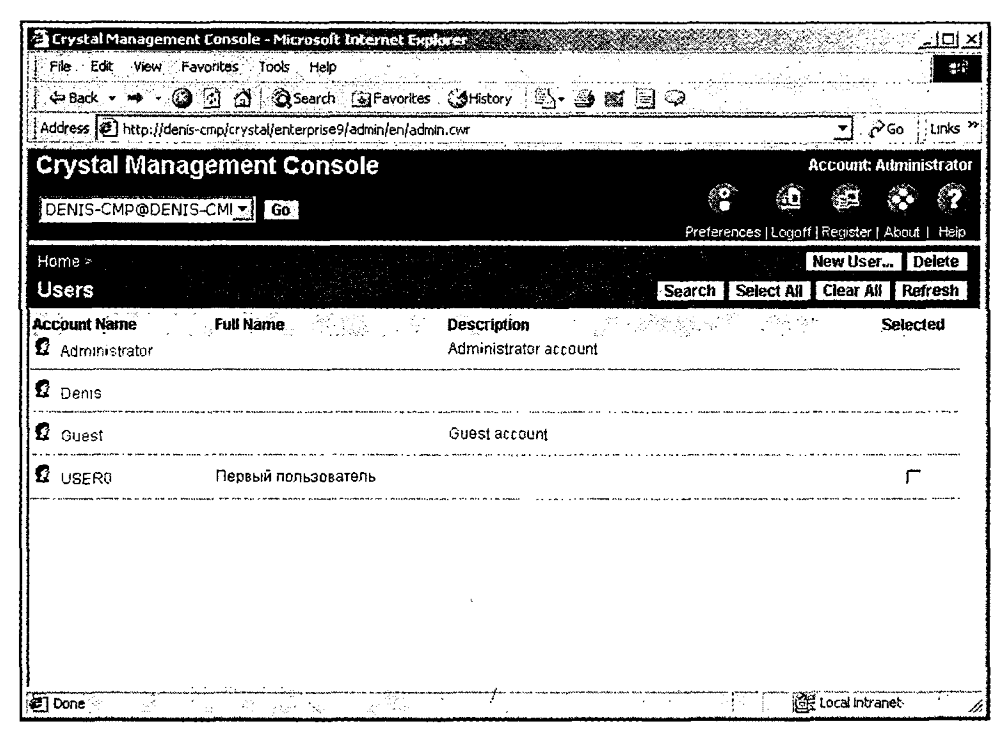


Рис. 8.19. Crystal Management Console. Страница со списком зарегистрированных пользователей

Для того чтобы зарегистрировать нового пользователя, необходимо нажать кнопку **New User**. При этом откроется страница **New User**, показанная на рис. 8.20. Заполнив поля в этой странице и установив соответствующие параметры, можно нажать кнопку **OK** для сохранения информации.

Рис. 8.20. **Crystal Management Console**. Регистрация нового пользователя

Обязательными к заполнению полями являются:

- **Account Name** — имя пользователя;
- **Password** — пароль пользователя;
- **Confirm** — подтверждение пароля.

Также возможна установка дополнительных параметров:

- **Connection Type** — тип пользовательского соединения:
  - Concurrent User — конкурентный пользователь;
  - Named User — именованный пользователь;
- **Password never expires** — действие пароля никогда не истекает;
- **User must change password at next logon** — пользователь должен изменить пароль при следующей регистрации;
- **User cannot change password** — пользователь не имеет права изменять пароль.

После выполнения регистрации нового пользователя или при редактировании прав существующего пользователя можно перейти в раздел **Member of** (рис. 8.21).

Рис. 8.21. Crystal Management Console. Ассоциация пользователя с группой

Рис. 8.22. Crystal Management Console. Страница **Groups**

Рис. 8.23. Crystal Management Console. Страница **New Group**

Рис. 8.24. Crystal Management Console. Страница **Administrators** раздел **Users**

Один пользователь может быть ассоциирован с одной или несколькими группами. Соответственно, такой пользователь будет обладать всеми теми привилегиями, которые определены у групп.

Для просмотра, редактирования и создания новых групп необходимо перейти на страницу **Manage Groups**, которая показана на рис. 8.22.

Чтобы создать новую группу пользователей, требуется нажать кнопку **New Group**. При этом откроется страница **New Group**, показанная на рис. 8.23.

После того как в поля **Group Name** и **Description** введена требуемая информация, можно нажать кнопку **OK** и перейти к странице **Users**, содержащей список ассоциированных с группой пользователей.

Нажав кнопку **Add Users**, можно ассоциировать новых пользователей с группой, выбрав их из списка. Если перейти на страницу **Subgroup**, то можно будет добавить подгруппы, которые так же, как пользователи, выбираются из списка. На странице **Member of** можно выполнить обратный процесс. То есть определить текущую группу как подгруппу в другой группе.

С помощью Crystal Management Console администратор имеет право быстро выполнять публикацию и настройку отчетов, действуя удаленно. Для этого необходимо перейти на страницу **Objects**, которая показана на рис. 8.25. На этой странице представлен список уже опубликованных отчетов с указанием папки, в которой они находятся, и с их описанием.

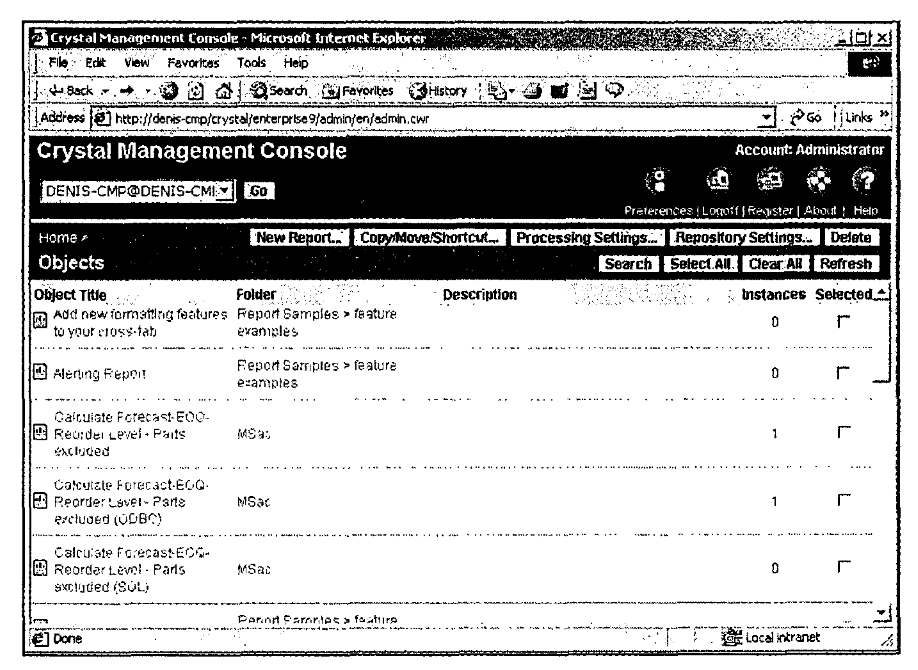


Рис. 8.25. Crystal Management Console. Страница **Objects**

Рис. 8.26. Crystal Management Console. Страница **New Report**

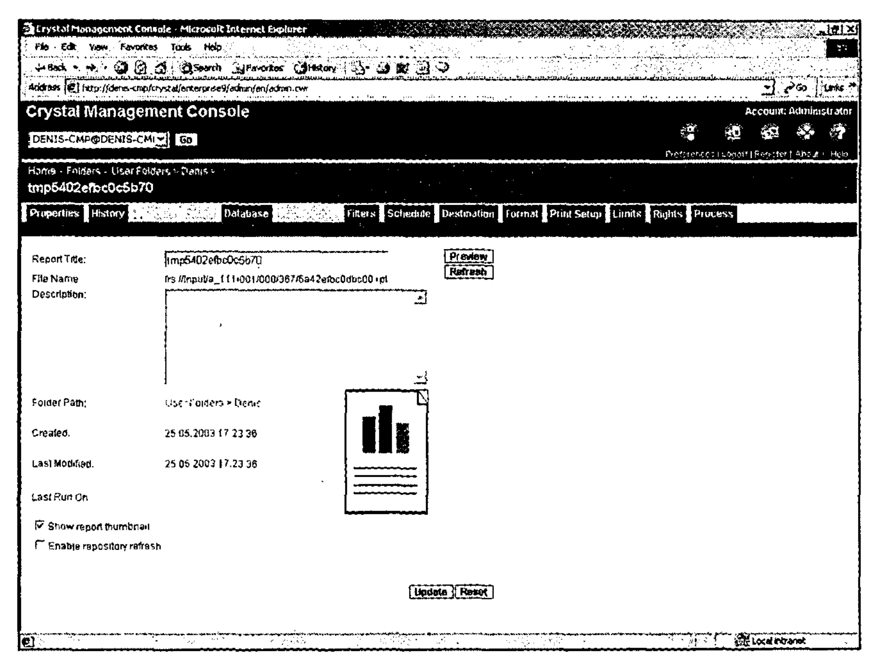


Рис. 8.27. Crystal Management Console. Страница с элементами настройки опубликованного отчета

Для публикации нового отчета необходимо нажать кнопку **New Report** и перейти на страницу **New Report** (рис. 8.26).

Нажав кнопку **Browse** с помощью стандартных механизмов операционной системы можно выбрать отчет, который требуется опубликовать. Далее указывается папка, в которой отчет должен быть сохранен. Папка может быть выбрана в группе полей, находящихся под комментарием **Then select a destination folder in which to save the report**. После того как отчет будет сохранен в указанной папке, внешний вид страницы с отчетом изменится так, как представлено на рис. 8.27.

На данной странице имеются следующие вкладки:

- **Properties** — основные параметры опубликованного отчета. Его системное имя, каталог, в который он записан, описание и т. п.;
- **History** — история работы с отчетом, если существует настройка его обработки по расписанию;
- **Alert Notification** — список сообщений, включенных в отчет;
- **Database** — параметры подключения отчета к базе данных;
- **Parameters** — список параметров, включенных в отчет;
- **Filters** — список условий фильтрации данных, включенных в запрос, формирующий набор данных для отчета;
- **Schedule** — параметры обработки отчета по расписанию;
- **Destination** — способ доставки отчета, полученного в результате его обработки по расписанию, до конечного пользователя;
- **Format** — формат, в котором доставляется отчет до конечного пользователя;
- **Print Setup** — настройки принтера для обработки отчета по расписанию, если назначена печать отчета;
- **Limits** — ограничения, накладываемые на хранимые экземпляры отчета;
- **Rights** — права, предоставленные различным группам на работу с опубликованным отчетом:
  - No Access — нет доступа;
  - View — только просмотр;
  - Schedule — только установка параметров расписания;
  - View On Demand — просмотр по требованию;
  - Full Control — полный доступ;
  - Advanced — развитые возможности работы с отчетом;
- **Process** — параметры, указывающие, на каком сервере и как необходимо выполнять данный отчет.

Выполнив все шаги настройки отчета, администратор может гарантировать корректную их работу и быструю доставку до конечного пользователя в требуемом виде.

Как видно из перечисленного, все отчеты хранятся в папках. Для создания папки необходимо перейти на страницу **Folders** и, нажав кнопку **New Folder**, перейти в окно **New Folder**, показанное на рис. 8.28.

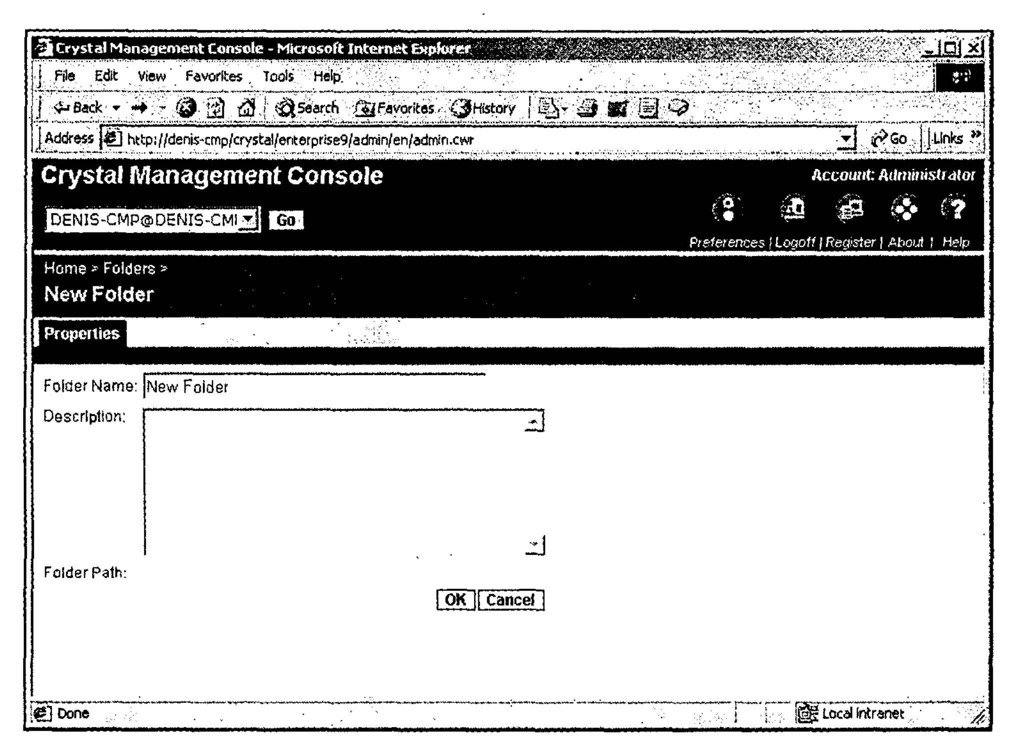


Рис. 8.28. Crystal Management Console. Страница **New Folder**

Определив имя папки и ее описание, можно сохранить данную папку в хранилище Crystal Enterprise. После сохранения папки станут доступными вкладки:

- **Reports** — список отчетов, которые опубликованы в данной папке. При необходимости можно добавить отчет, выбрав его из списка уже опубликованных отчетов;
- **Subfolders** — список подкаталогов;
- **Limits** — ограничения, наложенные на хранящиеся экземпляры отчета;
- **Rights** — привилегии на работу с отчетом, предоставленные различным группам пользователей.

К основным элементам администрирования относятся возможности по удаленному управлению и настройке компонентов самого Crystal Enterprise. Для того чтобы перейти к настройке данных компонентов, необходимо вызвать страницу **Manage Servers**.

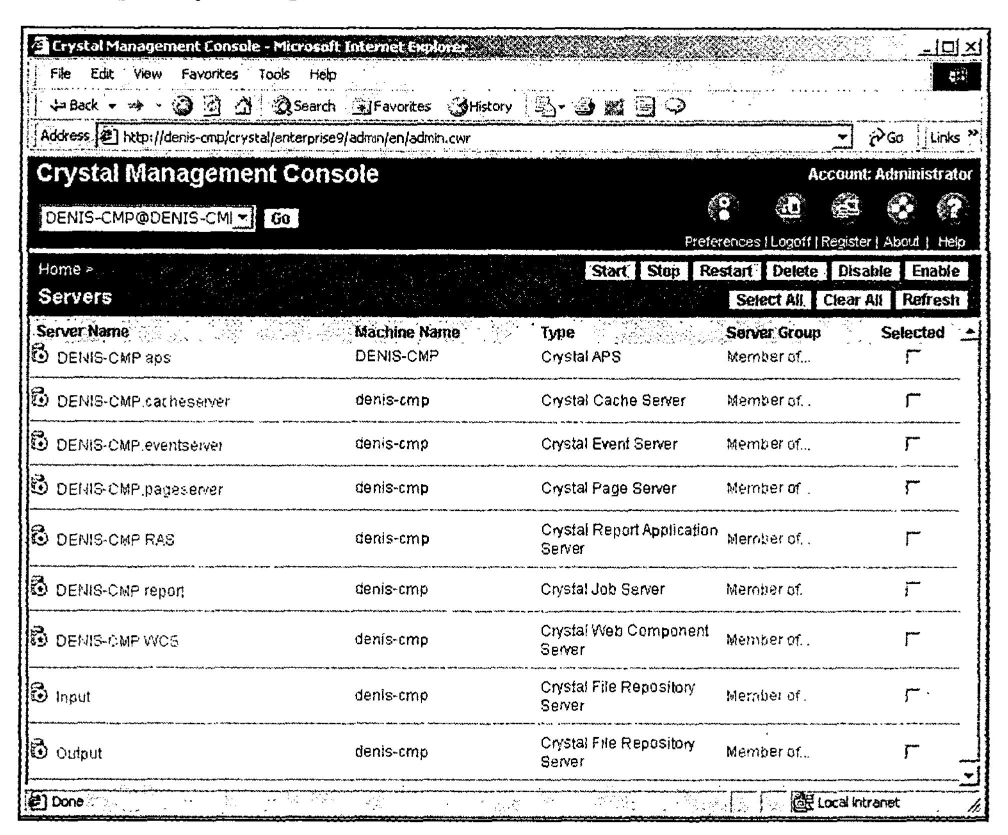


Рис. 8.29. Crystal Management Console. Страница управления серверами **Crystal Enterprise**

Как видно из рисунка, список компонентов, представленных на данной странице, полностью совпадает со списком компонентов, входящих в Crystal Management Console. Для того чтобы выполнить настройку необходимого компонента, требуется перейти по соответствующей ссылке на страницу редактирования параметров и произвести желаемые изменения. Необходимо помнить о том, что страничка **Servers** автоматически не обновляется. Так как после каждого изменения параметров выбранного компонента этот компонент перезапускается, для получения сведений о нормальной работе компонента потребуется перезагрузить страницу **Servers**.

Все остальные элементы администрирования относятся либо к информации только для чтения, либо позволяют выполнить настройки, связанные

с ограничениями, накладываемыми на отчеты или на пользователей. Все эти ограничения можно наложить и с помощью описанных ранее компонентов Crystal Management Console.

Очень часто возникает необходимость распространять отчеты не по одному, а целыми группами, выбирая их из каталога на компьютере. Такую функциональность предоставляет утилита Crystal Publishing Wizard. Вызов этого приложения можно выполнить с помощью команды меню **Start | Programs | Crystal Enterprise 9 | Crystal Publishing Wizard**. Появится окно, которое показано на рис. 8.30.

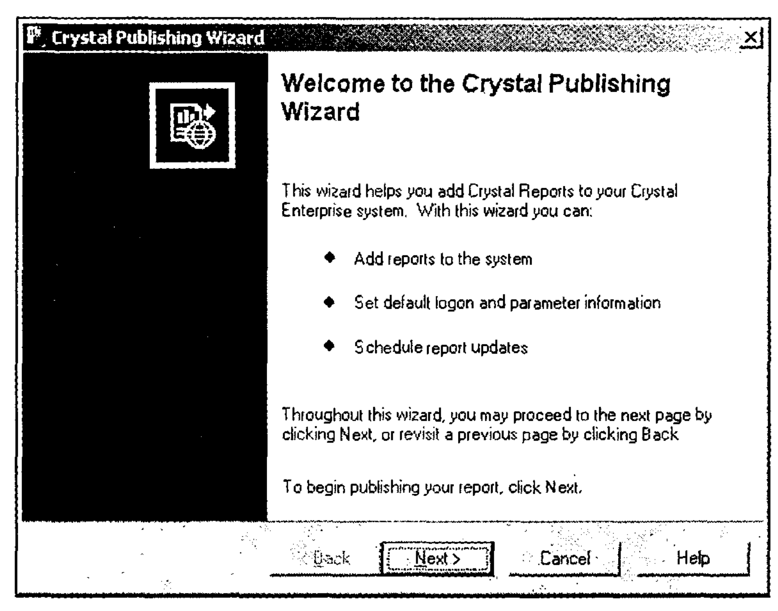


Рис. 8.30. Диалоговое окно **Crystal Publishing Wizard**

Утилита построена по принципу многостраничного диалога. При нажатии кнопки **Next** производится переход на следующую страницу. Следующей страницей, которая открывается за той, что изображена на рис. 8.30, является страница, позволяющая выбрать один или несколько отчетов для публикации.

В этом окне есть такие элементы:

- **Add Multiple Reports** — параметр, разрешающий одновременную публикацию нескольких отчетов;
- **Find Directory** — поиск каталога, содержащего отчеты для публикации, кнопка доступна только при активном параметре **Add Multiple Reports**;
- **Find File** — поиск файла отчета для публикации;

- **Add Directory** — добавление файлов отчетов из указанного каталога в список разрешенных к публикации, кнопка доступна только при активном параметре **Add Multiple Reports**;
- **Include Subfolders** — параметр, позволяющий добавить информацию из подкаталогов, доступен только при активном параметре **Add Multiple Reports**;
- **Select All** — выбор всех отчетов для публикации, кнопка доступна только при активном параметре **Add Multiple Reports**;
- **Deselect All** — отмена выбора всех отчетов для публикации, кнопка доступна только при активном параметре **Add Multiple Reports**.

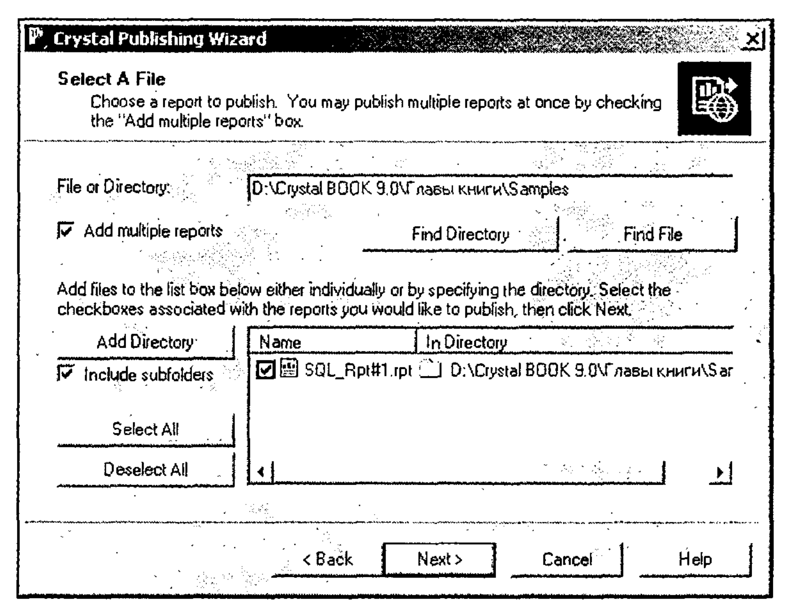


Рис. 8.31. Диалоговое окно **Crystal Publishing Wizard**. Выбор публикуемых отчетов

Следующим шагом после выбора отчетов для публикации будет регистрация на сервере Crystal Enterprise. На данном этапе поддерживаются те же механизмы, что и в Crystal Management Console. После регистрации в Crystal Enterprise появляется возможность выбора папки, в которую будут записаны выбранные отчеты, если требуемой папки не существует, ее можно создать.

Далее можно будет убедиться в том, что выбранные отчеты будут записаны в указанную папку. После того как администратор убедился в правильности указанного пути сохранения отчетов, он может перейти к установке параметров обработки отчетов по расписанию (рис. 8.33).

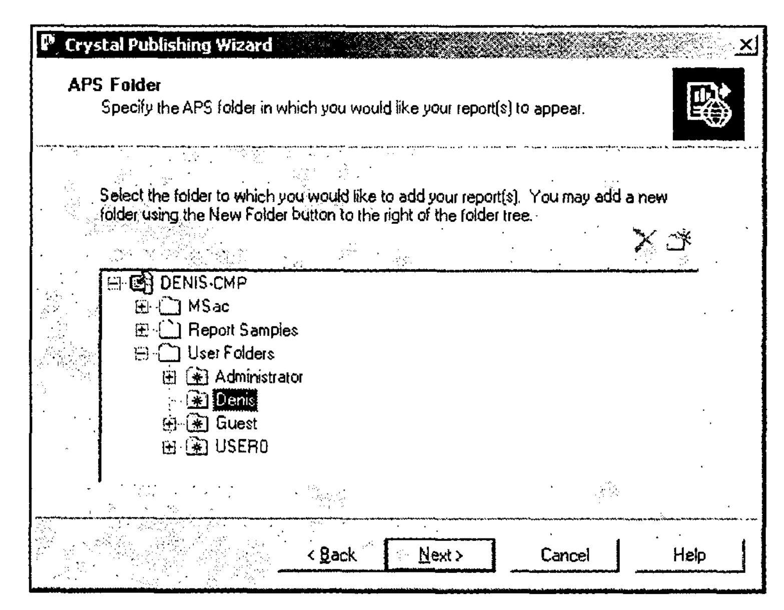


Рис. 8.32. Выбор папки, для сохранения отчетов

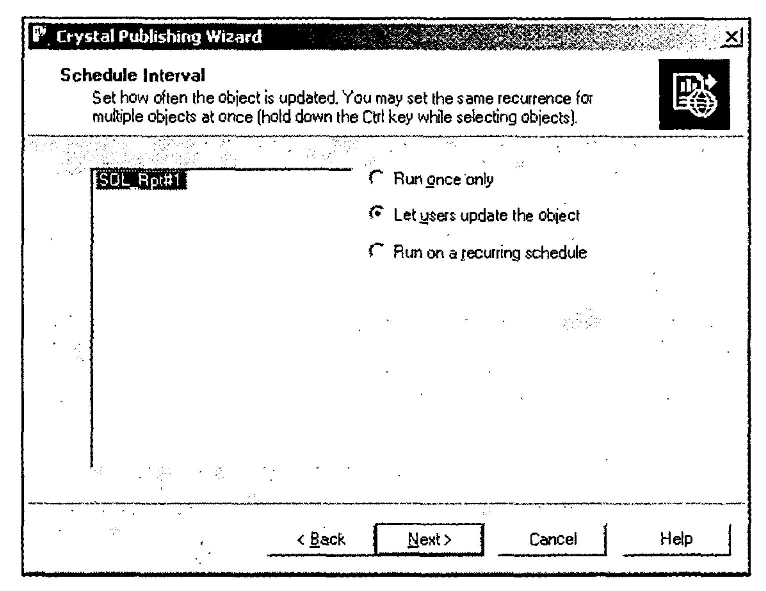


Рис. 8.33. Выбор принципа публикации отчета и его обработки по расписанию

В этом окне можно выбрать один из трех вариантов публикации отчета:

- **Run once only** — запустить отчет один раз на обработку сразу после публикации.

- **Let users update the object** — предоставить пользователю возможность задать параметры обработки самостоятельно.
- **Run on a recurring schedule** — установить параметры обработки по графику заранее.

Следующий шаг — выбор способа публикации.

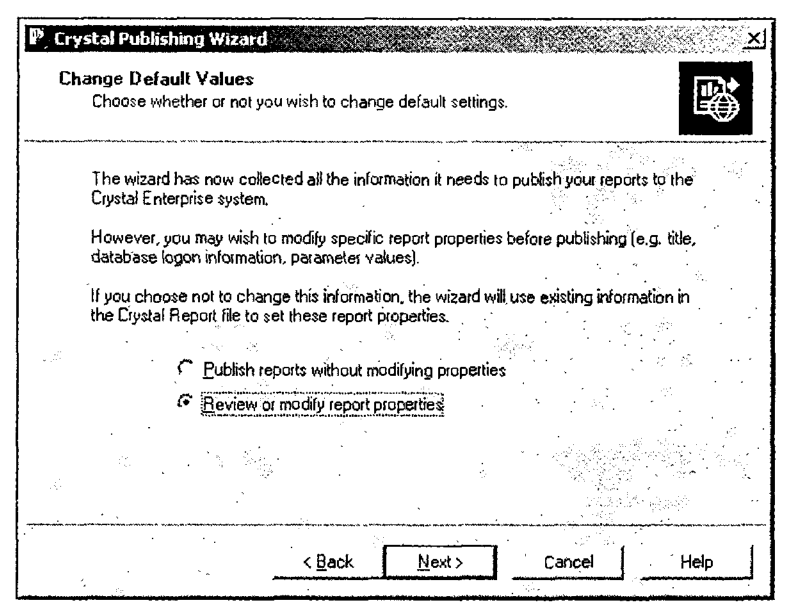


Рис. 8.34. Выбор варианта публикации

Есть два варианта:

- **Publish reports without modifying properties** — публикация отчетов без изменения их свойств.
- **Review or modify report properties** — просмотреть или изменить свойства отчетов.

В случае, когда выбран второй вариант публикации отчетов, появится возможность изменить наименование отчета, сделать его описание, определить параметры подключения к базе данных, задать значения параметров, включенных в отчет, и т. п.

Если выбран не один отчет, а несколько, надо будет определить параметры всех отчетов в списке. После того как заданы все параметры для выбранных отчетов, можно выполнять операцию их публикации. В случае, если в момент публикации не произошло никаких ошибок, будет получено сообщение, показанное на рис. 8.35.

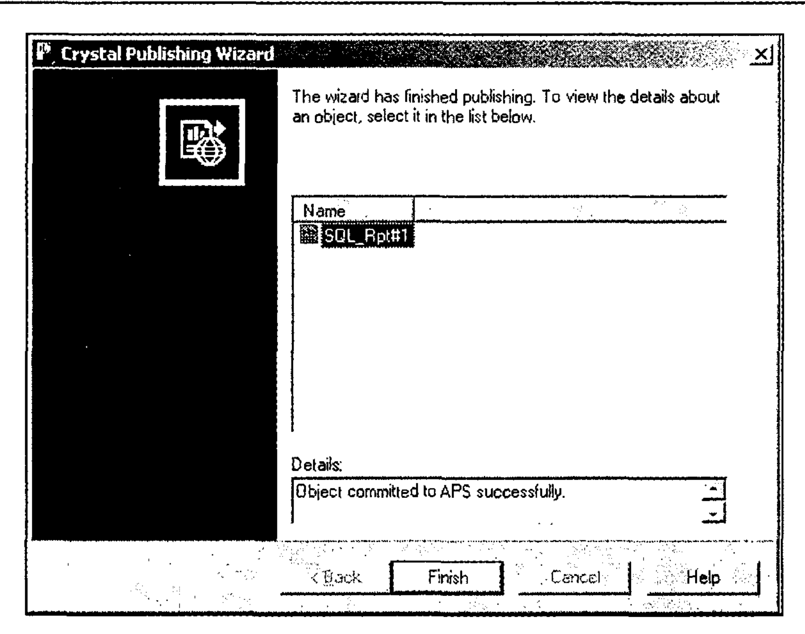


Рис. 8.35. Окно завершения публикации отчета или группы отчетов

Таким образом, используя данную утилиту, администратор Crystal Enterprise имеет возможность быстро предоставить конечным пользователям доступ к большому количеству новых отчетов, заранее определив ряд необходимых параметров.

Утилиты Crystal Import Wizard и Crystal Web Wizard используются крайне редко и имеют следующую функциональность:

- **Crystal Import Wizard** — позволяет выполнять импорт данных из более ранних версий Crystal Enterprise. Механизм импорта данных полностью совпадает с тем подходом, который используется при переключении на другой источник данных в качестве хранилища Crystal Enterprise;
- **Crystal Web Wizard** — позволяет создать немного другой внешний вид для рабочего места пользователя. Изменения, которые можно получить с помощью данной утилиты, относятся к цветовому оформлению, внешнему виду некоторых рабочих страниц **ePortfolio** и допустимым вариантам подключения к Crystal Enterprise. В большинстве случаев такие изменения быстрее выполняются на уже существующих страницах.

8.2.3. Рабочее место конечного пользователя — *ePortfolio*

Для того чтобы конечный пользователь имел возможность работать с отчетами, необходимо наличие всего лишь браузера (Internet Explorer или

Netscape Communicator). Вызвать рабочее окно пользователя **ePortfolio** можно одним из следующих способов:

- в браузере набрать команду следующего вида:
  - http://<Имя Web-сервера>/crystal/enterprise9/ePortfolio;
- если на компьютер конечного пользователя устанавливалась клиентская часть Crystal Enterprise, то обратиться к рабочему окну можно с помощью выполнения команды меню **Start | Programs | Crystal Enterprise 9 | Crystal Launchpad** и в появившемся окне выбрать ссылку **ePortfolio**.

И в том и в другом случае появится страница, показанная на рис. 8.36, предлагающая, как и в случае работы с Crystal Management Console, выбрать способ авторизации на сервере Crystal Enterprise и указать имя и пароль.

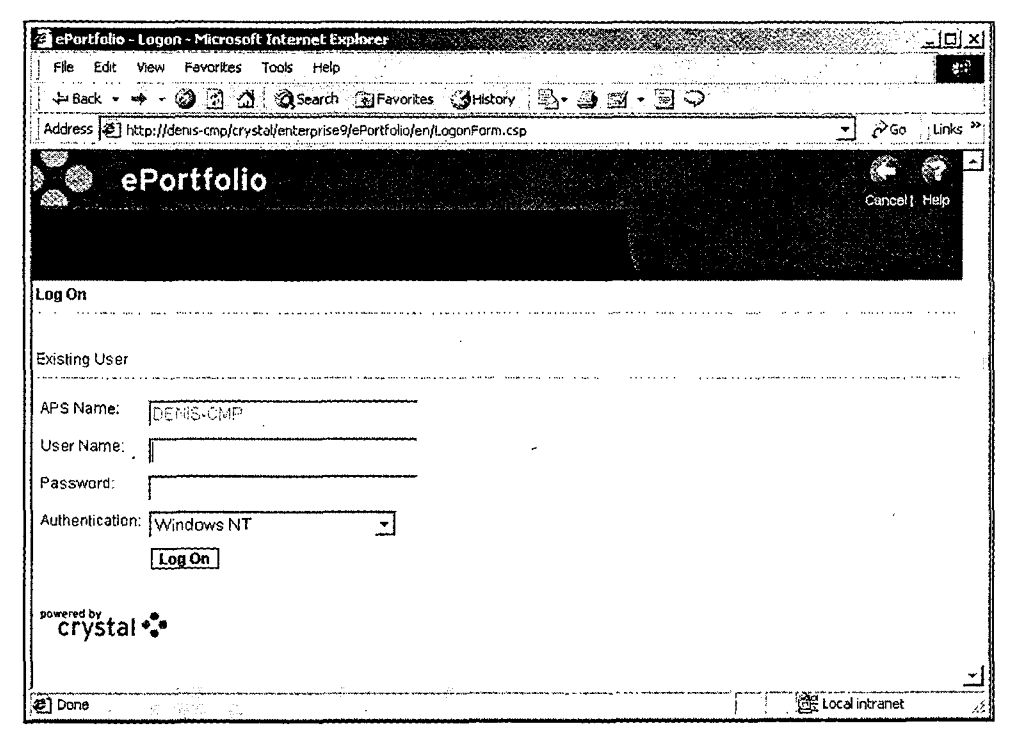


Рис. 8.36. ePortfolio — Log On

После того как все параметры заданы и нажата кнопка **Log On**, появляется страница, которая строго индивидуальна для каждого пользователя. На этой странице указано имя зарегистрированного пользователя, представлен список отчетов, с которыми пользователь может работать, и есть ряд элементов, позволяющих выполнять персональные настройки **ePortfolio** и организовывать поиск требуемых отчетов по различным условиям.

Рис. 8.37. Рабочее окно зарегистрированного пользователя Crystal Enterprise

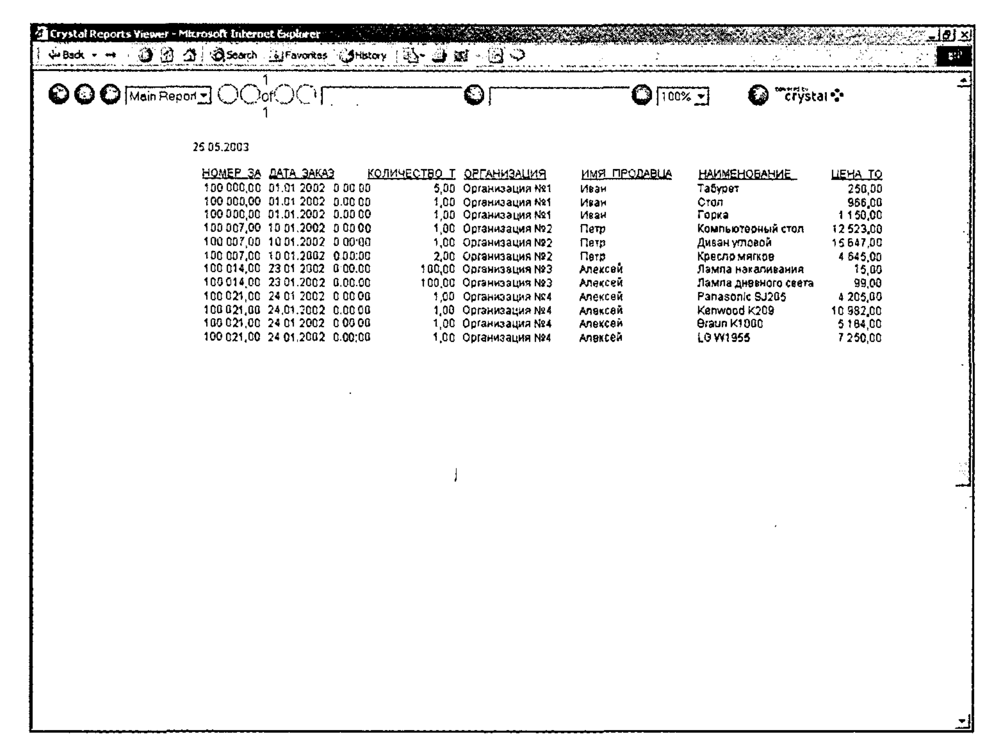
Рис. 8.38. Контекстное меню для отчета в ePortfolio

Для работы с отчетами есть контекстное меню, которое появляется при одинарном щелчке левой клавишей мыши на иконке отчета в рабочем окне (рис. 8.38).

Вызываемое меню имеет как минимум три команды:

- **View** — просмотр отчета в заранее заданном формате.
- **Schedule** — установка параметров расписания для обработки отчета.
- **History** — история по экземплярам отчета.

При вызове команды **View** на экране появляется новое окно браузера, в котором отображается либо отчет, либо предлагается ввести параметры подключения к базе данных и задать различные параметры, если они нужны. Отчет, отображенный в браузере, показан на рис. 8.39.



25.05.2003

НОМЕР_ЗА	ДАТА_ЗАКАЗ	КОЛИЧЕСТВО_Т	ОРГАНИЗАЦИЯ	ИМЯ_ПРОДАВЦА	НАИМЕНОВАНИЕ	ЦЕНА_ТО
100 000,00	01.01.2002 0:00:00	5,00	Организация №1	Иван	Табурет	250,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Стол	966,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Горка	1 150,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Компьютерный стол	12 523,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Диван угловой	15 647,00
100 007,00	10.01.2002 0:00:00	2,00	Организация №2	Петр	Кресло мягкое	4 645,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа накаливания	15,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа дневного света	99,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Panasonic SJ205	4 205,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Kenwood K209	10 982,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Braun K1000	5 184,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	LG W1955	7 250,00

Рис. 8.39. Отчет, просматриваемый с помощью Web-технологий

Как видно из рисунка, в отчет попадает не только необходимая для работы информация, но и включаются элементы навигации, позволяющие облегчить работу клиента с большим объемом данных.

На рис. 8.40 представлен вариант страницы, которая будет показана в случае необходимости определения параметров подключения к базе данных.

Подобного рода страница будет сформирована в случае наличия в отчете параметров и других, динамически задаваемых условий.

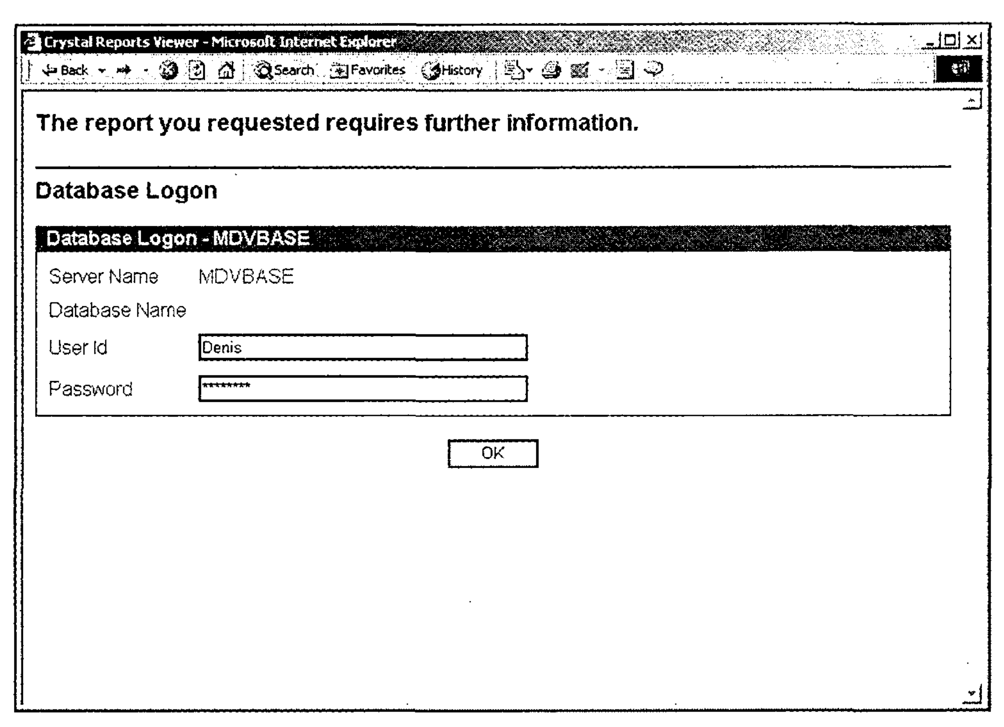


Рис. 8.40. Страница ввода параметров соединения с базой данных, используемой в отчете

Если вызвать команду **Schedule**, то на экране появится дополнительное окно браузера, в котором будет отображена страница настройки параметров обработки отчета по расписанию (рис. 8.41).

Как видно из рисунка, существуют разные варианты запуска отчета по времени:

- **Now** — запустить отчет непосредственно сейчас;
- **Once** — запустить отчет один раз в указанное время;
- **Hourly** — запускать отчет через каждые N часов;
- **Daily** — запускать отчет через каждые N дней;
- **Weekly** — запускать отчет в указанный день недели;
- **Monthly** — запускать отчет каждый N-ый месяц в году;
- **Nth Day of Month** — запускать отчет в указанный день каждого месяца;
- **1st Monday of Month** — запускать отчет в первый понедельник каждого месяца;

- **Last Day of Month** — запускать отчет в последний день каждого месяца;
- **X Day of Nth Week of the Month** — запускать отчет в указанный день указанной недели каждого месяца.

Рис. 8.41. Страница **Schedule**. Выбор времени запуска отчета

Рис. 8.42. Страница **Schedule**. Выбор других настроек для выполнения отчета по расписанию

Помимо временных параметров существует набор других параметров, которые можно установить для корректной обработки отчета. Список возможных настроек представлен на рис. 8.42.

К возможным настройкам относятся:

- **Schedule** — настройка параметров времени запуска, описанная ранее;
- **Database Logon** — настройка параметров доступа отчета к базе данных;
- **Parameters** — установка значений параметров, используемых в отчете;
- **Filters** — настройка параметров фильтрации данных, поступающих в отчет из источника;
- **Destination** — настройка параметров сохранения результата обработки отчета:
  - **Default** — сохранение в каталог, который определен при настройке Crystal Enterprise;
  - **Unmanaged Disk** — сохранение в каталог, который может находиться на любом доступном диске;
  - **Email (SMTP)** — отправка отчета по почте с использованием протокола SMTP;
  - **FTP** — размещение отчета на FTP-сайте;
- **Format** — формат, в котором необходимо получить результат от обработки отчета:
  - **Crystal Report** — стандартный формат отчета;
  - **Excel** — экспорт отчета в формат Excel;
  - **Excel (Data Only)** — экспорт в Excel только данных;
  - **Word** — экспорт отчета в формат MS Word;
  - **Acrobat** — экспорт отчета в формат PDF;
  - **Rich Text** — экспорт отчета в формат RTF;
  - **Plain Text** — экспорт отчета в текстовом формате;
  - **Paginated Text** — экспорт отчета в текстовый формат с разбивкой на страницы;
  - **Tab-separated Values** — экспорт отчета в текстовый формат с разделителями табуляцией;
  - **Character-separated Values** — экспорт отчета в текстовый формат с выбранными символьными разделителями;
- **Print Settings** — настройка параметров печати отчета после его обработки.

После установки параметров обработки отчета по расписанию и нажатия кнопки **Schedule** появляется окно **History** (рис. 8.43).

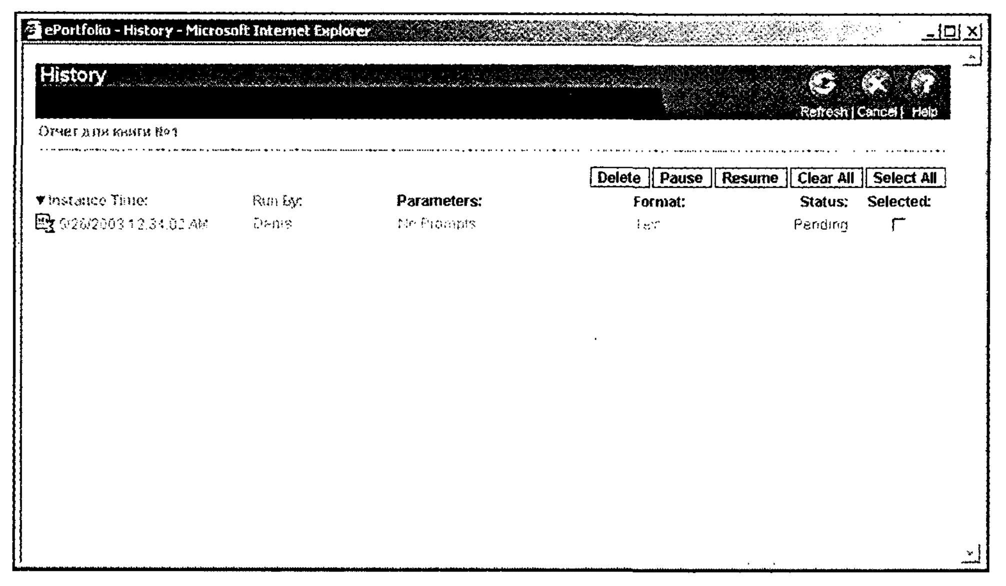


Рис. 8.43. Окно **Histrory**. Момент ожидания обработки отчета

Точно такое же окно **History** можно вызвать для любого доступного отчета, выполнив команду **History**. Единственным отличием такого окна в данном случае будет статус отчета:

- отчет не имеет истории. Окно пустое;
- отчет содержит список, состоящий из информации о выполнении отчета по расписанию:
  - Pending — отчет ожидает выполнения;
  - Running — отчет обрабатывается;
  - Success — отчет обработан корректно;
  - False — отчет обработан с ошибками.

В случае если отчет был обработан без ошибок, его можно открыть для просмотра. Если отчет обработан с ошибками, то информацию об ошибках можно узнать по ссылке **False**. Кроме того, после обработки отчета по расписанию в контекстном меню появляется дополнительная команда **View Latest Instance** (рис. 8.44).

При наличии большого количества опубликованных и доступных для работы отчетов иногда желательно выполнить поиск. В окне **ePortfolio** имеется механизм поиска отчетов по различным параметрам. Для того чтобы найти отчет, необходимо в поле **Look For** ввести требуемое условие поиска, в выпадающем списке выбрать критерии поиска и нажать кнопку **Search**. Если существует отчет, который удовлетворяет заданным критериям, он будет выведен на экран в виде соответствующей иконки.

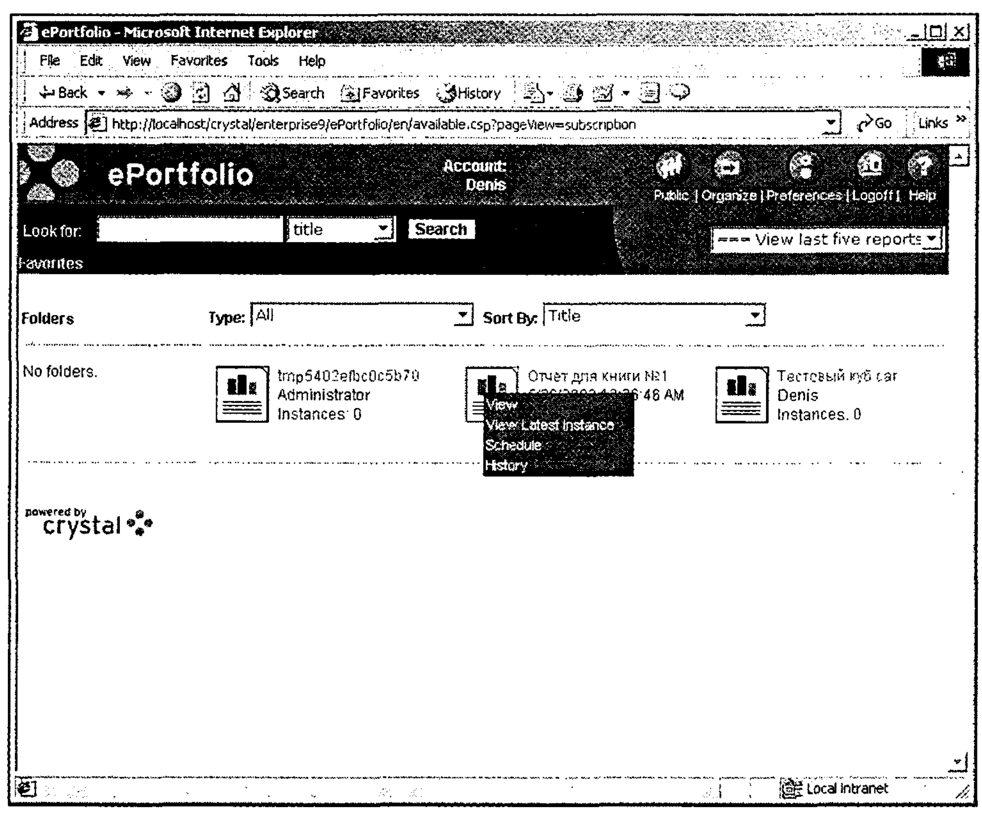


Рис. 8.44. Основное рабочее окно **ePortfolio**. Контекстное меню отчета, обработанного по расписанию

Критерии поиска могут быть следующими:

- **Title** — поиск по заголовку отчета;
- **Description** — поиск по описанию отчета;
- **folder title** — поиск по наименованию папки;
- **all fields** — поиск по всем возможным вариантам.

Помимо элементов работы с отчетами и средств поиска в **ePortfolio** есть ряд кнопок, при нажатии которых можно получить доступ к дополнительным возможностям:

- **Public/Favorites** — данная кнопка при нажатии меняет свой статус. Позволяет быстро переходить от общедоступных отчетов к наиболее часто используемым;
- **Organize** — помогает организовать структуру папок и отчетов, хранимых в этих папках;

- **Preferences** — предоставляет возможность настроить персональные параметры рабочего места **ePortfolio**;
- **Logoff** — выход из **ePortfolio**;
- **Help** — вызов подсказки.

При нажатии кнопки **Organize** открывается страница **Organize Folders**, показанная на рис. 8.45.

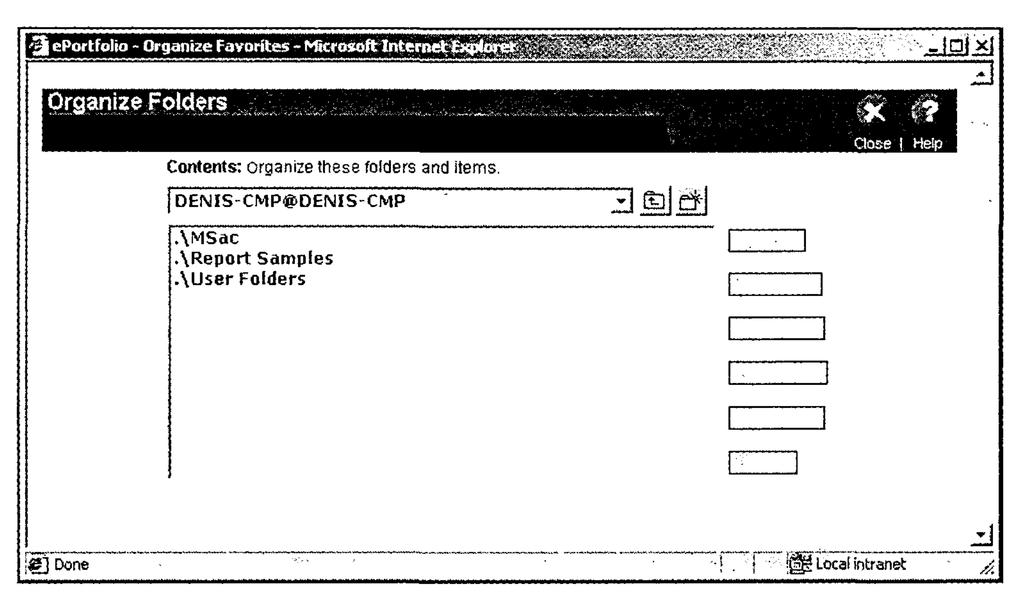


Рис. 8.45. Настройка параметров папок Crystal Enterprise

В данном окне есть следующая функциональность:

- **Expand** — раскрыть содержимое папки;
- **Copy To** — скопировать папку или отчет в указанное место;
- **Move To** — переместить папку или отчет в указанное место;
- **Shortcut** — организовать ссылку на отчет;
- **Rename** — переименовать папку или отчет;
- **Delete** — удалить папку или отчет.

При нажатии кнопки **Preferences** открывается страница **User Preferences**, на которой можно определить основные характеристики рабочего места клиента.

Возможности по настройке рабочего места таковы.

- **Initial View** — набор установок, позволяющий определить папку, показываемую в момент запуска **ePortfolio**.

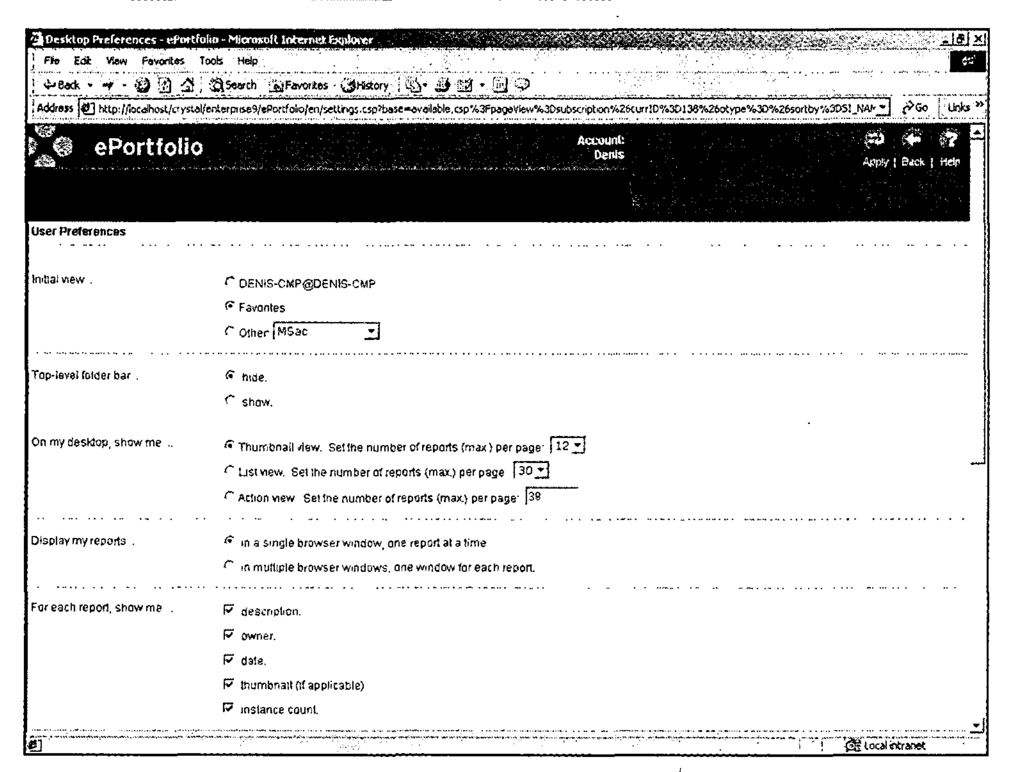


Рис. 8.46. Окно **User Preferences**

- **Top-level folder bar** — параметр, который позволяет скрыть или показать верхнюю панель кнопок в рабочих окнах.
- **On my desktop show me** — в рабочем окне показывать отчеты в следующем виде:
  - **Thumbnail view. Set the number of reports (max.) per page** — отчеты со снимками (если есть снимки). Выбирается количество отчетов на страницу.
  - **List view. Set the number of reports (max.) per page** — отчеты должны быть представлены в виде списка. Выбирается количество отчетов на страницу.
  - **Action view. Set the number of reports (max.) per page** — отчеты должны быть представлены в виде списка со ссылками на допустимые действия. Выбирается количество отчетов на страницу.
- **Display my reports**:
  - **in a single browser window, one report at a time** — одно окно для просмотра отчетов. Можно увидеть только один отчет;

- **in multiple browser windows, one window for each report** — каждый отчет открывается в своем окне.

- **For each report show me** — определяет, что показывать:
  - **Description** — описание;
  - **Owner** — владельца;
  - **Date** — дату публикации;
  - **thumbnail (if applicable)** — снимок, если существует;
  - **instance count** — количество готовых экземпляров.
- **Show menu as** — определяет, в каком виде показывать меню:
  - **buttons with text** — кнопки с текстом;
  - **buttons only** — только кнопки;
  - **text only** — только текст.
- **Display my desktop with this color scheme** — отображать рабочее окно с использованием цветовой схемы, представленной в таблице.
- **View my reports using the...** — просматривать отчеты с использованием одной из следующих технологий:
  - **ActiveX** — в качестве окна просмотра используется компонент ActiveX, требуется установка компонента на компьютер клиента;
  - **DHTML** — в формате HTML-страницы;
  - **Interactive DHTML** — просмотр отчета в формате HTML с рамками (Frame);
  - **Java** — просмотр отчета с помощью Java Applet, требуется установка компонента на компьютер клиента;
  - **Java Plug-In** — просмотр отчета выполняется с помощью компонента Java Plug-In, требуется установка компонента на компьютер клиента;
  - **Navigator Plug-In** — просмотр отчета выполняется с помощью компонента, аналогичного по своим характеристикам ActiveX, требуется установка компонента на компьютер клиента.
- **Preferred measuring units for report page layout is...** — предпочитаемые единицы измерения.
- **My current time-zone is...** — текущая временная зона. Используется для обработки отчетов по расписанию.

Различные механизмы просмотра отчетов в браузере позволяют подобрать тот вариант, который наиболее подходит к отчетам определенного вида. Единственным недостатком такого подхода является то, что на компьютер клиента необходимо установить ActiveX, Java, Java Plug-In или Navigator Plug-In. Не во всех компаниях клиент имеет право устанавливать какое-либо

Crystal Reports Viewer - Microsoft Internet Explorer

Main Report 1 of 1 100% crystal

26.05.2003

НОМЕР_ЗА	ДАТА_ЗАКАЗ	КОЛИЧЕСТВО_Т	ОРГАНИЗАЦИЯ	ИМЯ_ПРОДАВЦА	НАИМЕНОВАНИЕ	ЦЕНА_ТО
100 000,00	01.01.2002 0:00:00	5,00	Организация №1	Иван	Табурет	250,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Стол	966,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Горка	1 150,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Компьютерный стол	12 523,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Диван угловой	15 647,00
100 007,00	10.01.2002 0:00:00	2,00	Организация №2	Петр	Кресло мягкое	4 645,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа накаливания	15,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа дневного света	99,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Panasonic SJ205	4 205,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Kenwood K209	10 982,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Braun K1000	5 184,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	LG W1855	7 250,00

Рис. 8.47. Отчет в формате DHTML

Crystal Report Viewer - Microsoft Internet Explorer

100% 1 of 1 crystal

Preview

26.05.2003

НОМЕР_ЗА	ДАТА_ЗАКАЗ	КОЛИЧЕСТВО_Т	ОРГАНИЗАЦИЯ	ИМЯ_ПРОДАВЦА	НАИМЕНОВАНИЕ	ЦЕНА_ТО
100 000,00	01.01.2002 0:00:00	5,00	Организация №1	Иван	Табурет	250,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Стол	966,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Горка	1 150,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Компьютерный стол	12 523,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Диван угловой	15 647,00
100 007,00	10.01.2002 0:00:00	2,00	Организация №2	Петр	Кресло мягкое	4 645,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа накаливания	15,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа дневного света	99,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Panasonic SJ205	4 205,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Kenwood K209	10 982,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Braun K1000	5 184,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	LG W1955	7 250,00

Рис. 8.48. Отчет в формате ActiveX

9 Зак 1021

Crystal Reports Viewer - Microsoft Internet Explorer

powered by crystal Interactive Viewer

1 of 1

26.05.2003

НОМЕР ЗА	ДАТА ЗАКАЗ	КОЛИЧЕСТВО Т	ОРГАНИЗАЦИЯ	ИМЯ ПРОДАВЦА	НАИМЕНОВАНИЕ	ЦЕНА ТО
100 000,00	01.01.2002 0:00:00	5,00	Организация №1	Иван	Табурет	250,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Стол	966,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Горка	1 150,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Компьютерный стол	12 523,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Диван угловой	15 647,00
100 007,00	10.01.2002 0:00:00	2,00	Организация №2	Петр	Кресло мягкое	4 645,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа накаливания	15,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа дневного света	99,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Panasonic SJ205	4 205,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Kenwood K209	10 982,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Braun K1000	5 184,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	LG W1955	7 250,00

Рис. 8.49. Отчет в формате Interactive DHTML

Crystal Report Viewer - Microsoft Internet Explorer

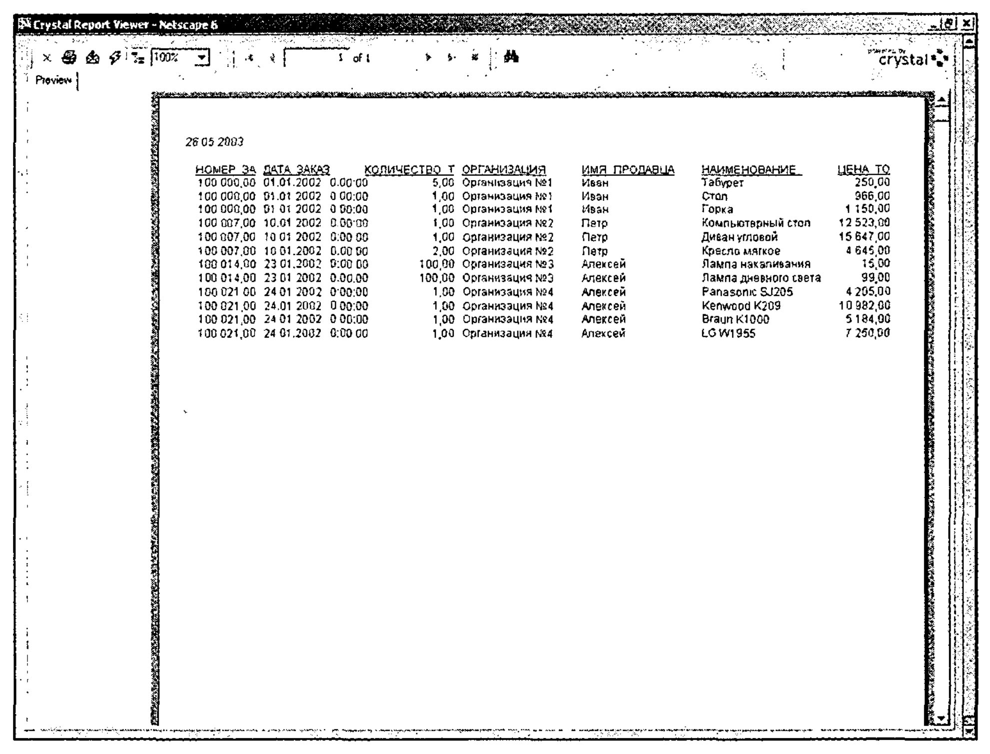
1 of 1 100% powered by crystal

Preview

26.05.2003

НОМЕР ЗА	ДАТА ЗАКАЗ	КОЛИЧЕСТВО Т	ОРГАНИЗАЦИЯ	ИМЯ ПРОДАВЦА	НАИМЕНОВАНИЕ	ЦЕНА ТО
100 000,00	01.01.2002 0:	5,00	Организация №1	Иван	Табурет	250,00
100 000,00	01.01.2002 0:	1,00	Организация №1	Иван	Стол	966,00
100 000,00	01.01.2002 0:	1,00	Организация №1	Иван	Горка	1 150,00
100 007,00	10.01.2002 0:	1,00	Организация №2	Петр	Компьютерный стол	12 523,00
100 007,00	10.01.2002 0:	1,00	Организация №2	Петр	Диван угловой	15 647,00
100 007,00	10.01.2002 0:	2,00	Организация №2	Петр	Кресло мягкое	4 645,00
100 014,00	23.01.2002 0:	100,00	Организация №3	Алексей	Лампа накаливания	15,00
100 014,00	23.01.2002 0:	100,00	Организация №3	Алексей	Лампа дневного све	99,00
100 021,00	24.01.2002 0:	1,00	Организация №4	Алексей	Panasonic SJ205	4 205,00
100 021,00	24.01.2002 0:	1,00	Организация №4	Алексей	Kenwood K209	10 982,00
100 021,00	24.01.2002 0:	1,00	Организация №4	Алексей	Braun K1000	5 184,00
100 021,00	24.01.2002 0:	1,00	Организация №4	Алексей	LG W1955	7 250,00

Рис. 8.50. Отчет в формате Java



26.05.2003

НОМЕР ЗА	ДАТА ЗАКАЗ	КОЛИЧЕСТВО Т	ОРГАНИЗАЦИЯ	ИМЯ ПРОДАВЦА	НАИМЕНОВАНИЕ	ЦЕНА ТО
100 000,00	01.01.2002 0:00:00	5,00	Организация №1	Иван	Табурет	250,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Стол	966,00
100 000,00	01.01.2002 0:00:00	1,00	Организация №1	Иван	Горка	1 150,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Компьютерный стол	12 523,00
100 007,00	10.01.2002 0:00:00	1,00	Организация №2	Петр	Диван угловой	15 647,00
100 007,00	10.01.2002 0:00:00	2,00	Организация №2	Петр	Кресло мягкое	4 645,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа накаливания	15,00
100 014,00	23.01.2002 0:00:00	100,00	Организация №3	Алексей	Лампа дневного света	99,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Panasonic SJ205	4 205,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Kenwood K209	10 982,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	Braun K1000	5 184,00
100 021,00	24.01.2002 0:00:00	1,00	Организация №4	Алексей	LG W1955	7 250,00

Рис. 8.51. Отчет в формате Navigator Plug-In

программное обеспечение без ведома администратора. Для того чтобы избавиться от необходимости установки перечисленных ранее компонентов, желательно пользоваться любым из вариантов DHTML. Этот механизм просмотра отчетов не потребует установки дополнительных компонентов.

Внешний вид отчетов, просматриваемых с помощью различных технологий, представлен на рис. 8.47—8.51. К сожалению, вариант просмотра отчетов с помощью Java Plug-In работает не во всех версиях браузеров, поэтому рисунка, соответствующего данному варианту, нет.

Таким образом, используя предложенную функциональность, пользователи быстро и качественно смогут настроить свое рабочее место и получать отчеты в том виде, который необходим для работы.

8.2.4. Локализация рабочего места конечного пользователя — *ePortfolio*

Стандартная поставка Crystal Enterprise предполагает использование английского языка в наименованиях компонентов рабочего окна пользователя. Однако очень часто перед компаниями, внедряющими Crystal Enterprise у себя

или на заказ, встает задача выполнить локализацию рабочих страниц **ePortfolio**. Учитывая то, что все основные рабочие страницы сформированы как CSP-страницы (Crystal Server Pages) — аналогом ASP (Active Server Pages), для людей, знакомых с ASP-технологией, не составит труда выполнить корректировку исходных текстов и получить требуемый результат. Кроме того, следует знать, что все CSP-страницы содержат специализированные разделы, предназначенные для локализации.

Примечание

Следует помнить, что все файлы CSP при редактировании необходимо сохранять в кодировке UTF8. Это связано с тем, что для работы с хранилищем и вывода информации в многоязыковом формате, такая кодировка подходит больше всего. Хотя в Crystal Enterprise 8.0 и Crystal Enterprise 8.5 существовал ряд проблем с этой кодировкой.

Все рабочие страницы Crystal Enterprise находятся в каталоге C:\Program Files\Crystal Decisions\Web Content\Enterprise9\ePortfolio\en\. С помощью обыкновенного текстового редактора можно открыть все файлы, находящиеся в указанном каталоге, и изменить значения соответствующих переменных. Различия между двумя файлами показаны в листингах 8.1 и 8.2.

Листинг 8.1. Часть файла available.csp до редактирования раздела, относящегося к локализации

```
<!--
      File Version Start - Do not remove this if you are modifying
      the file
      Build: 9.0.0
      File Version End
-->

<html>
<head>

<%@ language=JavaScript codepage=65001%>
<%
// available.csp
//
// This file is responsible for the main page of the application.
// It displays the list of folders and the list of objects
// in the current folder. It handles the public and
```

```
// favorites view, and searching.
//

//*******************************************************************
//SERVER-SIDE STRING VARIABLES FOR LOCALIZING
//*******************************************************************
var L_WELCOME_1 = "Account: ";
var L_REPORT_TITLE_SEARCHPARAMETER = "title";
var L_REPORT_DESCRIPTION_SEARCHPARAMETER = "description";
var L_SUBCATEGORY_TITLE_SEARCHPARAMETER = "folder title";
var L_ALL_FIELDS_SEARCHPARAMETER = "all fields";
var L_HOME = "Home ";
var L_HOME_SUBSCRIPTION_PATH = "Favorites";
var L_SEARCH_RESULTS_PATH = "Search Results: ";
var L_PUBLIC = "Public";
var L_MYREPORTS = "Favorites";
var L_SIGNUP = "Sign Up";
var L_LOGOFF = "Logoff";
var L_LOGON = "Log On";
var L_VIEWMANAGER = "Organize";
var L_EXIT = "Exit";
var L_SETTINGS = "Preferences";
var L_HELP = "Help";
var L_TYPE = "Type:";
var L_ALL = "All";
var L_REPORT = "Report";
var L_ANALYTICAL_REPORT = "Analytical Report";
```

Листинг 8.2. Часть файла available.csp после частичного редактирования раздела, относящегося к локализации

```
<!--
       File Version Start - Do not remove this if you are modifying
       the file
       Build: 9.0.0
       File Version End
-->

<html>
<head>
```

```
<%@ language=JavaScript codepage=65001%>

<%
// available.csp
//
// This file is responsible for the main page of the application.  It
displays the list
// of folders and the list of objects in the current folder.  It handles
the public and
// favorites view, and searching.
//

//*********************************************************************
//SERVER-SIDE STRING VARIABLES FOR LOCALIZING
//*********************************************************************
var L_WELCOME_1 = "Учетная запись: ";
var L_REPORT_TITLE_SEARCHPARAMETER = "заголовок отчета";
var L_REPORT_DESCRIPTION_SEARCHPARAMETER = "описание отчета";
var L_SUBCATEGORY_TITLE_SEARCHPARAMETER = "наименование папки";
var L_ALL_FIELDS_SEARCHPARAMETER = "все поля";
var L_HOME = "Home ";
var L_HOME_SUBSCRIPTION_PATH = "Favorites";
var L_SEARCH_RESULTS_PATH = "Search Results: ";
var L_PUBLIC = "Public";
var L_MYREPORTS = "Favorites";
var L_SIGNUP = "Sign Up";
var L_LOGOFF = "Logoff";
var L_LOGON = "Log On";
var L_VIEWMANAGER = "Organize";
var L_EXIT = "Exit";
var L_SETTINGS = "Preferences";
var L_HELP = "Help";
var L_TYPE = "Type:";
var L_ALL = "All";
var L_REPORT = "Report";
var L_ANALYTICAL_REPORT = "Analytical Report";
```

Результат обработки первого листинга показан на рис. 8.52.

Рис. 8.52. Окно **ePortfolio** без локализации

Рис. 8.53. Окно **ePortfolio**. Изменены значения некоторых переменных

Результат обработки второго листинга показан на рис. 8.53.

Таким образом, используя самый обыкновенный текстовый редактор, любой человек может выполнить локализацию всей рабочей области **ePortfolio** в сжатые сроки.



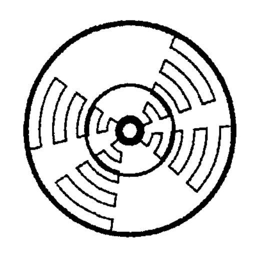
В случае если не все элементы изменяются при редактировании файлов CSP, необходимо обратиться к Web-разработчикам для уточнения места, где можно исправить наименование объекта, выводимое на экран.

Помимо описанных возможностей работы в Crystal Enterprise, есть также механизмы внесения дополнительной функциональности. Для того чтобы узнать, каким образом выполняется программирование и внесение своих собственных функций, необходимо обратиться к разделу документации Crystal Enterprise SDK, которая поставляется совместно с пакетом Crystal Enterprise.



Следует помнить, что вся основная и дополнительная функциональность должна быть реализована с помощью языка JavaScript. Это делается для поддержки многоплатформенности.

Глава 9



Интеграция отчетов, созданных с помощью Crystal Reports, в приложения

9.1. Возможности интеграции

Как видно из содержания предыдущих глав, Crystal Reports 9 является мощным и удобным генератором отчетов, а Crystal Enterprise предоставляет средства доставки отчетов пользователям. Однако очень часто возникает необходимость доставлять отчеты конечным пользователям не с помощью Web-технологий, а интегрировать имеющийся набор отчетов в приложения. Учитывая эту необходимость, компания Crystal Decisions (Seagate Software) предлагает несколько механизмов, позволяющих вызывать созданные с помощью Crystal Reports 9 отчеты из приложений, устанавливаемых у пользователей. В зависимости от квалификации разработчиков и времени, отведенного на разработку, можно выбирать из следующих возможностей:

- **Crystal Report Engine API** — наиболее развитый, но и наиболее трудоемкий механизм;
- **Crystal Report ActiveX Control** — самый простой и быстрый способ интеграции отчетов, но поддерживается не вся функциональность, доступная в **Crystal Report Engine API**;
- **Crystal Report Component для Delphi** — компонент, созданный на основе функций Crystal Reports API, поддерживаются различные версии Delphi;
- **Crystal Report RDC (Report Designer Component)** — компонент, позволяющий не только выполнять интеграцию существующих отчетов, но и создавать новые отчеты в момент работы приложения, требует дополнительного лицензирования.

Каждый из представленных механизмов обладает своими достоинствами и недостатками. Поэтому для того чтобы иметь возможность сравнить предлагаемые способы, рассмотрим несколько примеров интеграции отчетов в приложения. В качестве средств разработки будем использовать Borland Delphi 6.0 и Microsoft Visual Basic 6.0, а в качестве механизмов интеграции

рассмотрим Crystal Report API, Crystal Report ActiveX Control и Crystal Report Component.



Описание возможности интеграции отчетов Crystal Reports 9 в приложения можно найти в разделе документации Crystal Developers Help, поставляемой совместно с Crystal Reports 9. И хотя все примеры в документации и в этой книге ориентированы на использование Visual Basic и Delphi, можно также воспользоваться любыми средствами разработки, которые допускают использование API либо ActiveX.

9.2. Использование Crystal Report Engine API

Для того чтобы получить доступ к функциям Crystal Report Engine API, необходимо каким-то образом подключить соответствующую библиотеку к разрабатываемому приложению. Библиотекой, которая содержит функции Crystal Report Engine, является файл crpe32.dll. Этот файл устанавливается совместно с Crystal Reports 9, и найти его можно в каталоге C:\Program Files\Common Files\Crystal Decisions\2.0\bin, если операционная система установлена на диске С. В этом же каталоге находятся все основные файлы, необходимые для работы компонентов Crystal Reports 9. При интеграции отчетов в приложения многие из этих файлов должны входить в конечный пакет, устанавливаемый у пользователя. При разработке приложений с помощью Delphi или Visual Basic, для подключения библиотеки можно воспользоваться специально созданными файлами-модулями, в противном случае следует применять механизмы работы с функциями API.



Многие из приведенных далее примеров содержат функции и типы, список которых приведен в *Приложении 1* и *Приложении 2*.

9.2.1. Вызов отчетов Crystal Reports из приложений. Использование функций Crystal Reports API

При разработке приложения с помощью Delphi необходимо обеспечить возможность вызова функций Crystal Report API. В простейшем случае можно воспользоваться информацией, изложенной в *Приложении 1* и в *Приложении 2*, и описать самостоятельно прототипы требуемых функций, однако для ускорения этого процесса можно воспользоваться уже готовым файлом CRDelphi.pas, содержащим описание всех функций и типов. Данный файл находится на диске с дистрибутивом Crystal Reports 9 в каталоге \\Tools\Developers. В том же самом каталоге находится архив с примерами. Для того чтобы

использовать данный файл, его удобно скопировать в каталог, где будет сохранен проект приложения.



Файл CRDelphi.pas и примеры доступны только в дистрибутивах Crystal Reports 9 Developer или Advanced.

Листинг 9.1. Заголовок файла CRdelphi.pas

```
(*  Delphi Declarations for CRPE(32).DLL
**  ====================================
**
**  File      : CRDELPHI.PAS
**  Authors   : James Anderson, Andre Couturier, Frank Zimmerman
**  Date      : 14 Oct 1999
**  Purpose   : This file presents the API to the Crystal Reports
**              Print Engine DLL.  This file will compile and execute in
**               Delphi 1, 2, 3, 4 and 5.
**  Language : Delphi 1, 2, 3, 4 and 5.
**  Copyright (c) 1991-1999 Seagate Software Information Management Group, Inc.
**
** Notes:  This file will create a CRDELPHI.DCU when you use compile and build.
**         To use the CRDELPHI.DCU, you will need to add the following line
**         of code to your 'uses' section of your Delphi application
**            eg.
**            uses
**               Classes, CRDELPHI, ...;
**
** UPDATES on November 23rd, 2000
** Added by Trevor Dubinsky
**
** Added declarations, types and constants required for following function calls:
** function PEGetReportVersion
** function PEGetNReportAlerts
** function PEGetNthReportAlert
** function PEGetNthAlertinstanceInfo
** function PESetNthAlertConditionFormula;
** function PESetNthAlertMessageFormula;
** function PESetNthAlertDefaultMessage;
```

```
** function PEEnableNthAlert;
*)
```

При разработке приложения с помощью Visual Basic можно поступить так же, как и при работе в Delphi. Только для импорта списка функций и типов к проекту необходимо подключить два модуля Global32.bas и Crwrap.bas, которые находятся в том же самом каталоге, что и CRDelphi.pas.

Листинг 9.2. Заголовок файла Global32.bas

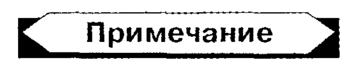
```
'
'              Visual Basic Declarations of CRPE32.DLL
'              =====================================
'
'       File:          GLOBAL32.BAS
'
'       Author:        Seagate Software Information Management Group, Inc.
'       Date:          15 Apr 92
'
'       Purpose:       This file presents the API to the Crystal Reports
'                            Print Engine DLL (Professional).
'
'       Language:      Visual Basic for Windows
'
'       Copyright (c) 1992 - 1999 Seagate Software Information Management Group, Inc.
'
```

Листинг 9.3. Заголовок файла Crwrap.bas

```
'
'              Visual Basic Declarations of crwrap32.DLL
'              =====================================
'
'       File:          crwrap.BAS
'
'       Author:        Seagate Software Information Management Group, Inc.
'
'
'       Language:      Visual Basic for Windows
```

```
'       Copyright (c) 1992 - 1999 Seagate Software Information Management Group, Inc.
'
'       Revisions:
'       SM- Nov 99, removed obsolete exporting formats
'           Excel 2.1
'            Global Const crUXFXls2Type% = 0
'           Excel 3.0
'            Global Const crUXFXls3Type% = 1
'           Excel 4.0
'            Global Const crUXFXls4Type% = 2
'           HTML 3.0
'            Global Const crUXFHTML3Type% = 0   ' Draft HTML 3.0 tags
'
'

'=======================================================
' Constants and Function Definitions for Exporting
'=======================================================
```



После подключения указанных файлов к проекту необходимо откорректировать файлы. В тех строчках, которые содержат приблизительный путь к библиотекам crpe32.dll и crwrap.dll, необходимо указать действительный путь. Если этого не сделать, то приложение работать не будет.

После подключения указанных файлов к проектам Delphi и Visual Basic, можно приступить к реализации приложения. Для начала желательно создать экранную форму, как это показано на рис. 9.1 и 9.2.

Для удобства работы установим параметры экранной формы, как в табл. 9.1 или в табл. 9.2.

Таблица 9.1. *Установка параметров экранной формы в проекте Delphi*

Элемент	Наименование	Заголовок
Экранная форма	CRW_API_Form	Использование Crystal Report Engine API
Кнопка	btnShowReport	Показать отчет
Диалог. Компонент OpenDialog	ReportsOpenDialog	Открыть отчет Crystal Report

Рис. 9.1. Проект и экранная форма Delphi

Рис. 9.2. Проект и экранная форма Visual Basic

У компонента типа OpenDialog с именем ReportsOpenDialog желательно установить фильтр. Для этого необходимо установить фокус на этот компонент в экранной форме и, перейдя в редактор **Object Inspector**, дважды щелкнуть левой клавишей мыши в поле с названием **Filter**. При этом появится диалоговое окно, в которое необходимо ввести значения так, как это показано на рис. 9.3.

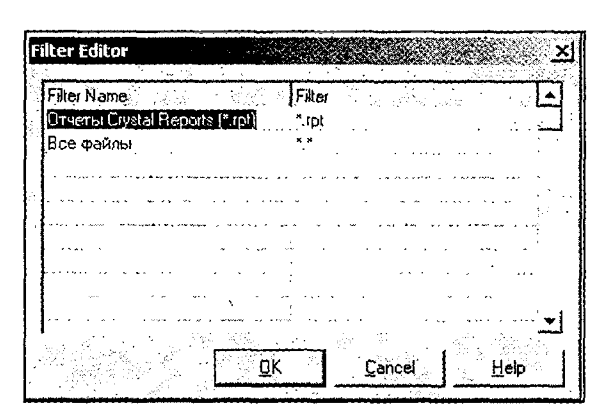


Рис. 9.3. Настройка параметров фильтрации выбираемых файлов Delphi

Таблица 9.2. *Установка параметров экранной формы в проекте Visual Basic*

Элемент	Наименование	Заголовок
Экранная форма	CRW_API_Form	Использование Crystal Report Engine API
Кнопка	btnShowReport	Показать отчет
Диалог. Компонент CommonDialog	ReportOpenDialog (CommonDialog)	Открыть отчет Crystal Report

У компонента типа CommonDialog с именем ReportOpenDialog так же, как и при работе в Delphi, желательно установить значение фильтра. Для этого необходимо вызвать контекстное меню компонента и выполнить команду **Properties**. В открывшемся диалоговом окне **Property Pages** на вкладке **Open/Save As** вписать информацию так, как это показано на рис. 9.4.

Далее можно приступить к описанию алгоритма вызова отчетов Crystal Report 9 нажатием кнопки **Показать отчет**. В обоих примерах компоненты ReportOpenDialog служат для выбора отчета, который необходимо открыть.



Все функции Crystal Report Engine API именуются по общему принципу. Первые две буквы наименования функции или процедуры всегда PE, например, PEOpenEngine. Описание функций можно найти в *Приложении 1*.

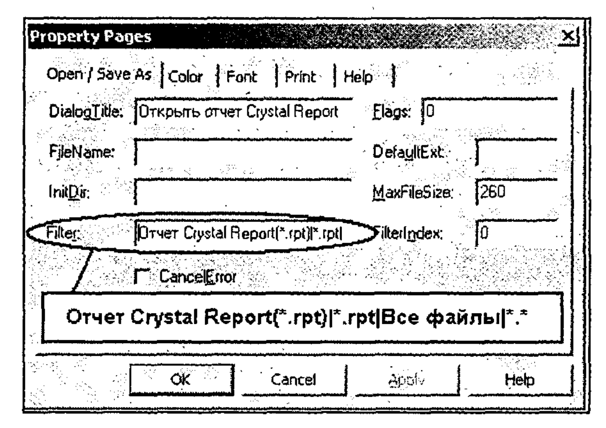


Рис. 9.4. Настройка фильтрации выбираемых файлов Visual Basic

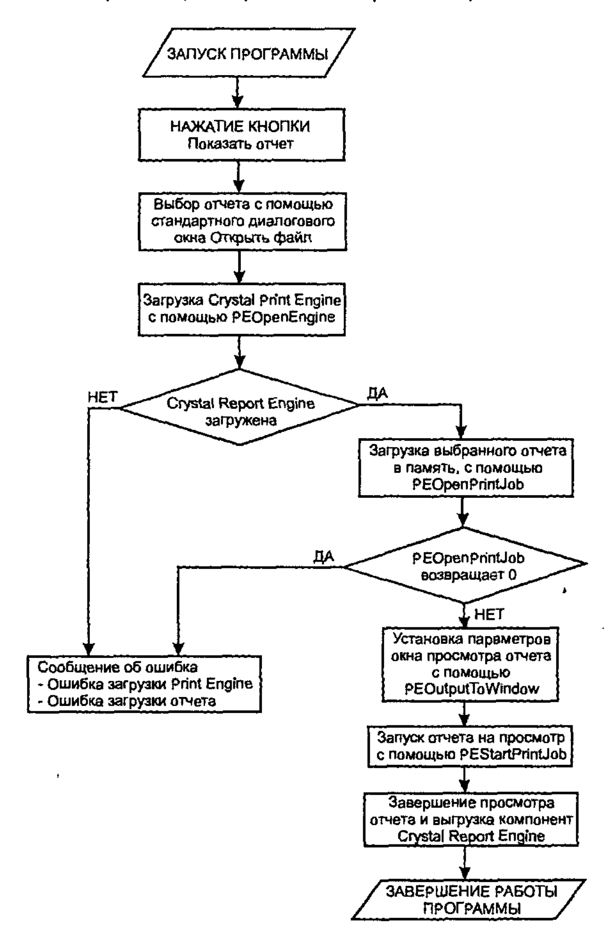


Схема 9.1. Алгоритм просмотра отчетов Crystal Reports 9 из приложения

Для удобства работы рекомендуется в разделе глобальных переменных объявить следующую переменную:

- **Для проекта Delphi** — nPrintJob, тип данных Word;
- **Для проекта Visual Basic** — nPrintJob, тип данных Integer.

В данном примере будет использовано всего четыре функции:

- **PEOpenEngine** — загрузка компонентов Crystal Print Engine;
- **PEOpenPrintJob** — загрузка в память выбранного отчета;
- **PEOutputToWindow** — настройка параметров окна просмотра отчетов;
- **PEStartPrintJob** — запуск отчета на просмотр.

Блок-схема алгоритма работы с отчетами представлена на схеме 9.1.

В листингах 9.4 и 9.5 дан текст процедур, реализующих алгоритм, представленный на схеме 9.1.

Листинг 9.4. Проект в Delphi, содержащий описание основных объектов и процедур

```
unit MainForm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
    TCRW_API_Form = class(TForm)
    btnShowReport: TButton;
    ReportsOpenDialog: TOpenDialog;
    procedure btnShowReportClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  CRW_API_Form: TCRW_API_Form;
  nPrintJob: Word; {Переменная, в которую присваивается номер обрабаты-
ваемого отчета}
```

```
implementation

uses CRDelphi;

{$R *.dfm}

procedure TCRW_API_Form.btnShowReportClick(Sender: TObject);
begin
 If ReportsOpenDialog.Execute Then
   Begin
   {Попытка запуска Crystal Reports Print Engine.
   Если запуск не удался, то выдается сообщение об ошибке.}
   If PEOpenEngine Then
   Begin
    {При корректном запуске Crystal Reports Print Engine
    выполняется попытка загрузить выбранный отчет.
    Если не удается загрузить отчет, то выдается сообщение об ошибке.}
    nPrintJob:=PEOpenPrintJob(PChar(ReportsOpenDialog.FileName));
    If nPrintJob=0 Then
     ShowMessage('Ошибка открытия отчета')
    Else
     Begin
     {Установка базовых характеристик окна просмотра отчета.}
     PEOutputToWindow(nPrintJob,
                     PChar(ReportsOpenDialog.FileName), 0,0,0,0,
     WS_MAXIMIZE+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_SYSMENU, 0);
     {Комментарий - PEOutputToWindow(Значение, возвращаемое функцией
                       PEOpenPrintJob,
                       Заголовок открываемого окна просмотра,
                       Координата X верхнего левого угла, Координата Y
                       верхнего левого угла,
                       Ширина окна просмотра, Высота окна просмотра,
                       Стиль окна просмотра (флаги могут быть перечислены
                       через "OR"),
                       Указатель на родительское окно);}
     {Запуск на просмотр выбранного отчета с определенными параметрами.}
     PEStartPrintJob(nPrintJob,TRUE);
     end
   end
 Else
```

```
  ShowMessage('Ошибка запуска CR Report Engine');
 end;
end;
end.
```

Листинг 9.5. Проект в Visual Basic, реализующий алгоритм со схемы 9.1

```
Dim nPrintJob As Integer ' Переменная, которой присваивается номер обра-
батываемого отчета

Private Sub btnShowReport_Click()
Dim result%

ReportOpenDialog.ShowOpen
If ReportOpenDialog.fileName <> "" Then
   'Попытка запуска Crystal Reports Print Engine.
   'Если запуск не удался, то выдается сообщение об ошибке.
   If PEOpenEngine Then
       'При корректном запуске Crystal Reports Print Engine
       'выполняется попытка загрузить выбранный отчет.
       'Если не удается загрузить отчет, то выдается сообщение об ошибке.
        nPrintJob = PEOpenPrintJob(ReportOpenDialog.fileName)
        If nPrintJob = 0 Then
            MsgBox ("Ошибка открытия отчета")
        Else
           'Установка базовых характеристик окна просмотра отчета.
           result% = PEOutputToWindow(nPrintJob,
          ReportOpenDialog.fileName, 0, 0, 0, 0,
          WS_MAXIMIZE + WS_MINIMIZEBOX + WS_MAXIMIZEBOX + WS_SYSMENU, 0)

           'Запуск на просмотр выбранного отчета с определенными
            параметрами.
           result% = PEStartPrintJob(nPrintJob, True)
        End If
   Else
        MsgBox ("Ошибка запуска CR Report Engine")
   End If
End If
End Sub
```

После отладки и компиляции представленного в листингах кода можно запустить полученные приложения и посмотреть, что получилось. В момент запуска внешний вид экранных форм должен быть таким, как показано на рис. 9.5 и 9.6.

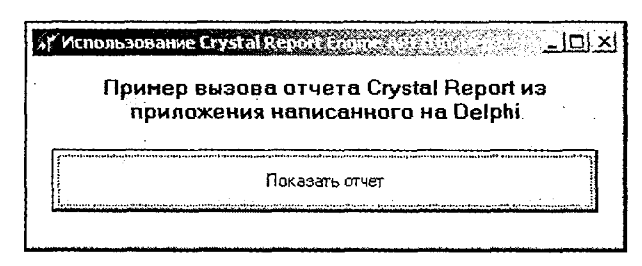


Рис. 9.5. Приложение, реализованное с помощью Delphi

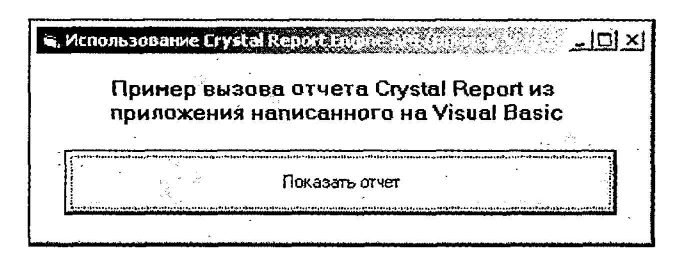


Рис. 9.6. Приложение, реализованное с помощью Visual Basic

Внешне обе экранные формы абсолютно идентичны, и различия между программами минимальны. Это говорит о том, что обе программы должны работать одинаково. При нажатии кнопки **Показать отчет** в любом из приложений появится стандартный диалог операционной системы, предлагающий выбрать файл отчета для отображения в окне просмотра. После выбора файла и нажатия кнопки **Открыть** (**Open**) результат будет выглядеть так, как это показано на рис. 9.7.



Результат будет аналогичен в случае, если в качестве среды разработки использовались не Delphi или Visual Basic, а любое другое средство разработки.

Упражнение 9.1

Создайте на основании описанных примеров программу в Delphi, Visual Basic или другом средстве разработки приложений и посмотрите результат.

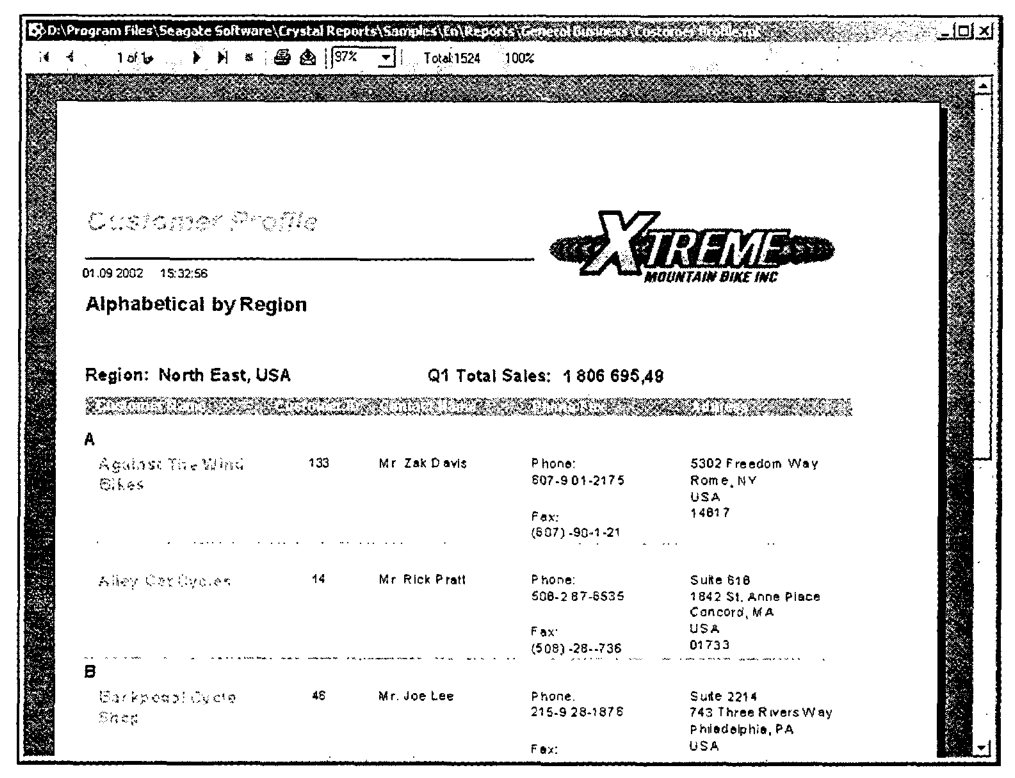


Рис. 9.7. Отчет Crystal Report 9, открытый в стандартном окне просмотра с помощью приложения, написанного в Delphi или Visual Basic

9.2.2. Вызов отчетов Crystal Reports из приложений. Настройка параметров окна просмотра отчетов с помощью функций Crystal Report API

Предыдущие примеры показали, как можно с помощью функций Crystal Report 9 Engine API выбрать любой отчет и открыть его для просмотра и дальнейшей печати. Однако не всегда удобно работать с отчетами в стандартном окне просмотра, содержащем минимум функциональности и не всегда дающем возможность использовать ряд специфических характеристик самого отчета. К наиболее важным из таких характеристик относятся:

- наличие группирования и возможность навигации по дереву групп;
- создание Drill Down отчетов;
- экспорт;
- поиск.

Для того чтобы иметь возможность управлять настройками окна просмотра отчетов необходимо внести ряд изменений в алгоритм и созданное ранее приложение. Блок-схема алгоритма представлена на схеме 9.2.

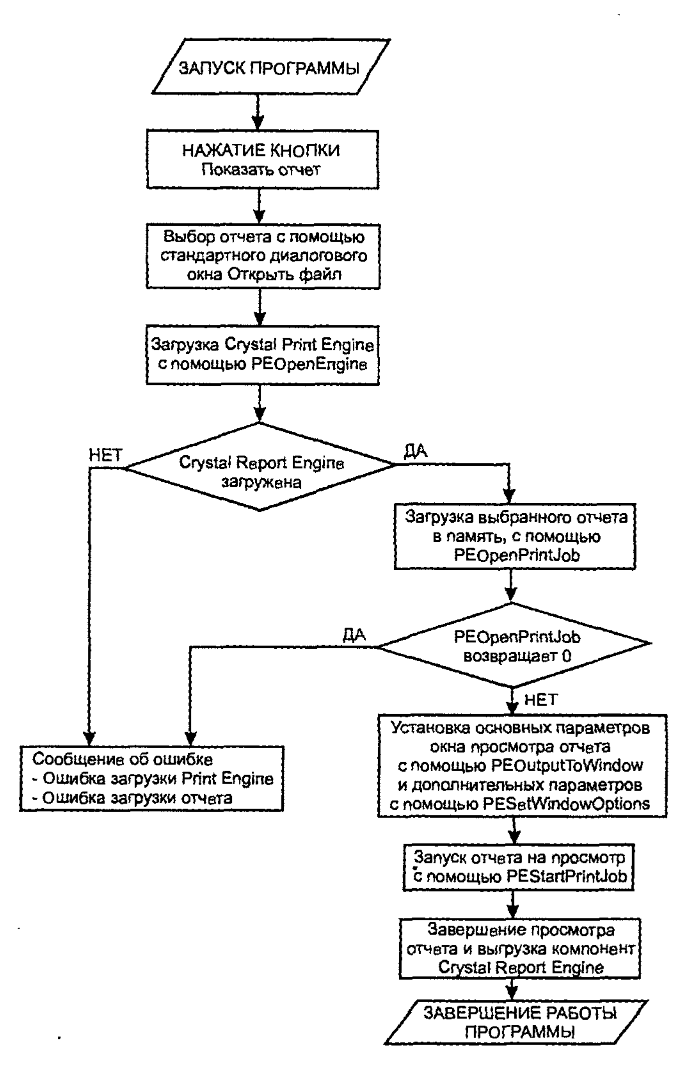


Схема 9.2. Блок-схема алгоритма, позволяющего произвести настройку параметров окна просмотра

Как видно из блок-схемы, в алгоритм добавилась еще одна функция — PESetWindowsOptions, которая потребует внесения дополнительных элементов управления в ранее описанные примеры. Во-первых, потребуется изменить внешний вид экранной формы, добавив 12 элементов типа CheckBox.

Эти элементы позволят устанавливать параметры окна просмотра отчета непосредственно перед выбором самого отчета. Экранные формы будут выглядеть так, как это показано на рис. 9.8 и 9.9.

Рис. 9.8. Внешний вид экранной формы для выбора параметров окна просмотра отчета в Delphi

Рис. 9.9. Внешний вид экранной формы для выбора параметров окна просмотра отчета в Visual Basic

Для удобства использования представленных далее листингов добавленные элементы интерфейса желательно назвать так, как это описано в табл. 9.3.

Таблица 9.3. *Настройка параметров экранной формы в проекте Delphi*

Тип объекта	Название	Текст, видимый на экране (Caption)
CheckBox	cbGroupTree	Дерево групп
CheckBox	cbDrillDown	Drill Down
CheckBox	cbNavigation	Элементы навигации
CheckBox	cbFormat	Форматирование
CheckBox	cbRefresh	Обновить
CheckBox	cbSearch	Поиск
CheckBox	cbCancel	Отмена
CheckBox	cbClose	Закрыть
CheckBox	cbProgress	Отображение прогресса загрузки
CheckBox	cbExport	Экспорт
CheckBox	cbPrint	Печать
CheckBox	cbPrinterSetup	Настройка печати

Перечисленные в таблице элементы для Delphi и Visual Basic идентичны.

После модификации формы можно приступать к изменению текста программы. Для установки параметров окна просмотра отчета нам потребуется вызов функции PESetWindowOptions и, поскольку один из параметров ее имеет тип PEWindowOptions, необходимо декларировать переменную ViewerOption этого типа. Далее потребуется проверка состояния каждого элемента типа CheckBox и присвоение соответствующих значений элементам переменной ViewerOption. Текст программы будет выглядеть так, как это показано в листингах 9.6 и 9.7.

Листинг 9.6. Отредактированная программа в Delphi, реализующая алгоритм просмотра отчетов в настраиваемом окне просмотра. Цветом выделены внесенные в предыдущий пример изменения

```
unit MainForm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
```

```
type
  TCRW_API_Form = class(TForm)
    btnShowReport: TButton;
    ReportsOpenDialog: TOpenDialog;
    Label1: TLabel;
    GroupBox1: TGroupBox;
    cbGroupTree: TCheckBox;
    cbDrillDown: TCheckBox;
    cbNavigation: TCheckBox;
    cbFormat: TCheckBox;
    cbRefresh: TCheckBox;
    cbSearch: TCheckBox;
    cbCancel: TCheckBox;
    cbClose: TCheckBox;
    cbProgress: TCheckBox;
    cbExport: TCheckBox;
    cbPrint: TCheckBox;
    cbPrinterSetup: TCheckBox;
    procedure btnShowReportClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  CRW_API_Form: TCRW_API_Form;
  nPrintJob: Word; {Переменная-указатель на выполняемый отчет}

implementation

uses CRDelphi;

{$R *.dfm}

procedure TCRW_API_Form.btnShowReportClick(Sender: TObject);
var
        ViewerOption: PEWindowOptions; {Переменная, используемая для
        установки дополнительных параметров окна просмотра отчетов}
```

```
begin
 If ReportsOpenDialog.Execute Then
   Begin
   {Попытка запуска Crystal Reports Print Engine.
   Если не удалось, то выдается сообщение об ошибке.}
   If PEOpenEngine Then
   Begin
    {При корректном запуске Crystal Reports Print Engine
    выполняется попытка загрузить выбранный отчет.
    Если не удается загрузить отчет, то выдается сообщение об ошибке.}
    nPrintJob:=PEOpenPrintJob(PChar(ReportsOpenDialog.FileName));
    If nPrintJob=0 Then
     ShowMessage('Ошибка открытия отчета')
    Else
     Begin
     {Установка базовых характеристик окна просмотра отчета.}
     PEOutputToWindow(nPrintJob,
                      PChar(ReportsOpenDialog.FileName), 0,0,0,0,
     WS_MAXIMIZE+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_SYSMENU, 0);
     {Комментарий - PEOutputToWindow(Значение, возвращаемое функцией
                       PEOpenPrintJob,
                       Заголовок открываемого окна просмотра,
                       Координата X верхнего левого угла, Координата Y
                       верхнего левого угла,
                       Ширина окна просмотра, Высота окна просмотра,
                       Стиль окна просмотра(флаги могут быть перечислены
                       через "OR"),
                       Указатель на родительское окно);}
     {Проверка состояния CheckBox-ов и
     настройка параметров окна просмотра отчета}
     If cbGroupTree.Checked Then
          ViewerOption.hasGroupTree := 1
     else
        ViewerOption.hasGroupTree := 0;
     If cbDrillDown.Checked Then
        ViewerOption.canDrillDown :=1
     else
        ViewerOption.canDrillDown :=0;
     If cbNavigation.Checked Then
        ViewerOption.hasNavigationControls :=1
     else
        ViewerOption.hasNavigationControls :=0;
```

```
If cbCancel.Checked Then
   ViewerOption.hasCancelButton :=1
else
   ViewerOption.hasCancelButton :=0;
If cbPrint.Checked Then
   ViewerOption.hasPrintButton :=1
else
   ViewerOption.hasPrintButton :=0;
If cbExport.Checked Then
   ViewerOption.hasExportButton :=1
else
   ViewerOption.hasExportButton :=0;
If cbFormat.Checked Then
   ViewerOption.hasZoomControl :=1
else
   ViewerOption.hasZoomControl :=0;
If cbClose.Checked Then
   ViewerOption.hasCloseButton :=1
else
   ViewerOption.hasCloseButton :=0;
If cbProgress.Checked Then
   ViewerOption.hasProgressControls :=1
else
   ViewerOption.hasProgressControls :=0;
If cbSearch.Checked Then
   ViewerOption.hasSearchButton :=1
else
   ViewerOption.hasSearchButton :=0;
If cbPrinterSetup.Checked Then
   ViewerOption.hasPrintSetupButton :=1
else
   ViewerOption.hasPrintSetupButton :=0;
If cbRefresh.Checked Then
   ViewerOption.hasRefreshButton :=1
else
   ViewerOption.hasRefreshButton :=0;
ViewerOption.StructSize := PE_SIZEOF_WINDOW_OPTIONS; {Константа,
                                                      равная 32}
{Передача заданных характеристик в окно просмотра отчета}
```

```
    PESetWindowOptions(nPrintJob,ViewerOption);

    {Запуск на просмотр выбранного отчета с определенными параметрами.}
    PEStartPrintJob(nPrintJob,TRUE);
    end
  end
 Else
  ShowMessage('Ошибка запуска CR Report Engine');
 end;
end;
end.
```

Листинг 9.7. Отредактированная программа в Visual Basic, реализующая алгоритм просмотра отчетов в настраиваемом окне просмотра. Цветом выделены внесенные в предыдущий пример изменения

```
Dim nPrintJob As Integer 'Переменная-указатель на выполняемый отчет

Private Sub btnShowReport_Click()
Dim result%
Dim ViewerOptions As PEWindowOptions 'Переменная, используемая для
        установки дополнительных параметров окна просмотра отчетов

ReportOpenDialog.ShowOpen
If ReportOpenDialog.fileName <> "" Then
   'Попытка запуска Crystal Reports Print Engine.
   'Если не удалось, то выдается сообщение об ошибке.
   If PEOpenEngine Then
       'При корректном запуске Crystal Reports Print Engine
       'выполняется попытка загрузить выбранный отчет.
       'Если не удается загрузить отчет, то выдается сообщение об ошибке.
        nPrintJob = PEOpenPrintJob(ReportOpenDialog.fileName)
        If nPrintJob = 0 Then
            MsgBox ("Ошибка открытия отчета")
        Else
           'Установка базовых характеристик окна просмотра отчета.
           result% = PEOutputToWindow(nPrintJob,
           ReportOpenDialog.fileName, 0, 0, 0, 0,
           WS_MAXIMIZE + WS_MINIMIZEBOX + WS_MAXIMIZEBOX + WS_SYSMENU, 0)
```

```
'Проверка состояния CheckBox-ов и
'настройка параметров окна просмотра отчета
If cbGroupTree Then
 ViewerOptions.hasGroupTree = 1
Else
 ViewerOptions.hasGroupTree = 0
End If
If cbDrillDown Then
 ViewerOptions.canDrillDown = 1
Else
 ViewerOptions.canDrillDown = 0
End If
If cbNavigation Then
 ViewerOptions.hasNavigationControls = 1
Else
 ViewerOptions.hasNavigationControls = 0
End If
If cbFormat Then
 ViewerOptions.hasZoomControl = 1
Else
 ViewerOptions.hasZoomControl = 0
End If
If cbRefresh Then
 ViewerOptions.hasRefreshButton = 1
Else
 ViewerOptions.hasRefreshButton = 0
End If
If cbSearch Then
 ViewerOptions.hasSearchButton = 1
Else
 ViewerOptions.hasSearchButton = 0
End If
If cbCancel Then
 ViewerOptions.hasCancelButton = 1
Else
 ViewerOptions.hasCancelButton = 0
End If
If cbClose Then
 ViewerOptions.hasCloseButton = 1
```

```
        Else
         ViewerOptions.hasCloseButton = 0
        End If
        If cbProgress Then
         ViewerOptions.hasProgressControls = 1
        Else
         ViewerOptions.hasProgressControls = 0
        End If
        If cbExport Then
         ViewerOptions.hasExportButton = 1
        Else
         ViewerOptions.hasExportButton = 0
        End If
        If cbPrint Then
         ViewerOptions.hasPrintButton = 1
        Else
         ViewerOptions.hasPrintButton = 0
        End If
        If cbPrinterSetup Then
         ViewerOptions.hasPrintSetupButton = 1
        Else
         ViewerOptions.hasPrintSetupButton = 0
        End If
        ViewerOptions.StructSize = PE_SIZEOF_WINDOW_OPTIONS
        result% = PESetWindowOptions(nPrintJob, ViewerOptions)

    'Запуск на просмотр выбранного отчета с определенными параметрами.
        result% = PEStartPrintJob(nPrintJob, True)
      End If
  Else
      MsgBox ("Ошибка запуска CR Report Engine")
  End If
End If
End Sub
```

Как видно из приведенных листингов, для реализации механизма передачи параметров окна просмотра, необходимо внести минимальные изменения. Большее время будет затрачено на выполнение чисто механической работы по внесению большого количества строк. После изменения текста можно

запустить полученную программу и посмотреть результаты, которые будут получаться при активизации одного или нескольких CheckBox. Результат вызова отчета после активирования всех CheckBox показан на рис. 9.10.

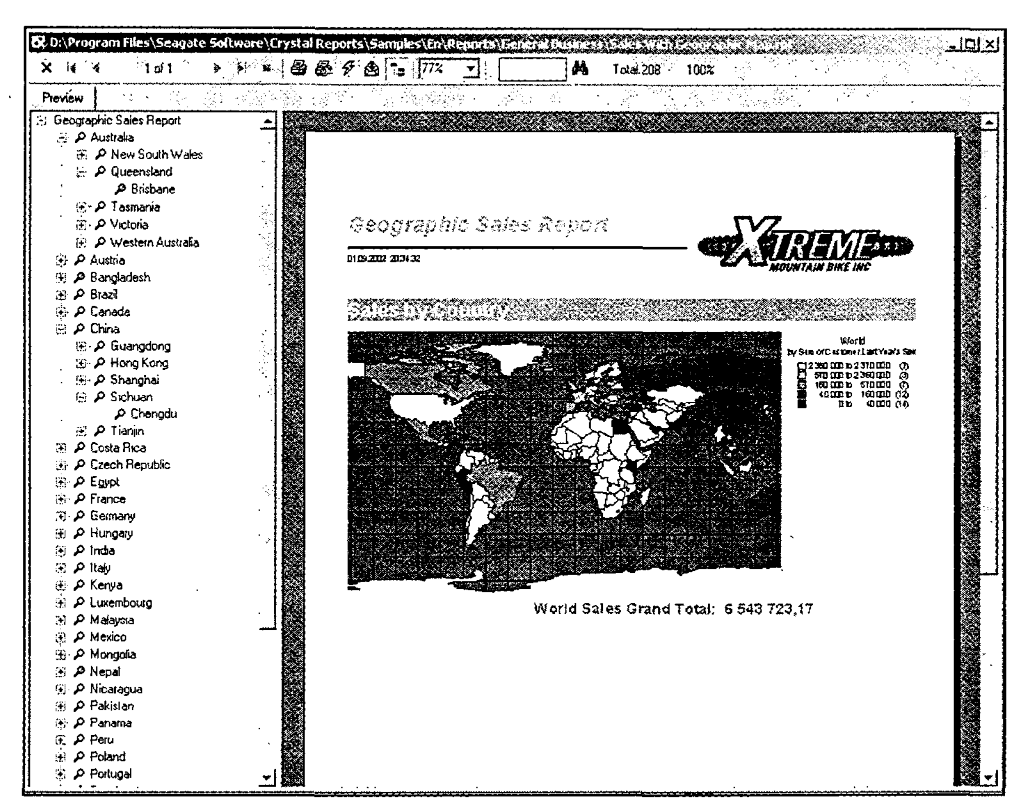


Рис. 9.10. Окно просмотра отчета, содержащее все доступные элементы управления

Упражнение 9.2

Внесите необходимые изменения в написанное ранее приложение и посмотрите результат.

9.2.3. Обработка отчетов с параметрами. Использование Crystal Report API

Часто при работе с отчетами обнаруживается, что вызываемый отчет должен содержать один или несколько параметров. Для того чтобы эти параметры можно было обрабатывать непосредственно в приложении, а не в момент загрузки отчета, существует большой набор функций. Работу некоторых из них мы рассмотрим в предлагаемых примерах.

Чтобы рассмотреть один из вариантов обработки отчета с параметром, необходимо создать отчет на основании данных из тестовой базы xtreme.mdb. Выполните следующие действия:

1. Подключитесь к базе данных xtreme.mdb.
2. Выберите таблицу **Customer**.
3. В таблице выберите поля Customer Name, Contact Title, Contact First Name, Contact Last Name и Last Year's Sales.
4. Отформатируйте полученный отчет.
5. Создайте поле параметр с названием "Условие группировки", типом данных String, подсказкой "Условие группировки. Впишите в поле букву: C — группировка по стране, R — группировка по региону, S — группировка по городу" и значениями по умолчанию — C, R и S так, как это показано на рис. 9.11 и 9.12.
6. Создайте поле формулу, под названием "Формула группировки". Текст ее представлен в листинге 9.8.
7. Сгруппируйте отчет по созданному полю формуле.
8. Результат должен быть похож на вариант, показанный на рис. 9.13. При каждом повторном запуске этого отчета или при выполнении операции обновления данных будет появляться диалоговое окно, предлагающее изменить значение параметра.

Листинг 9.8. Текст формулы

```
if {?Условие группировки}="C" then {Customer.Country}
    else if {?Условие группировки}="R" then {Customer.Region}
        else if {?Условие группировки}="S" then {Customer.City}
```

Рис. 9.11. Диалоговое окно **Edit Parameter Field**

Set Default Values

Select from database

Browse table:

Browse field:

Select or enter value to add.

Default Values — Description

C
R
S

Import pick list — Export pick list — Define description

Options

Length limit

Min Length: 0

Max Length: 0

Edit mask:

Display: Value and description

Order: No sort

Order based on: Value

OK — Cancel — Help

Рис. 9.12. Диалоговое окно **Set Default Values**

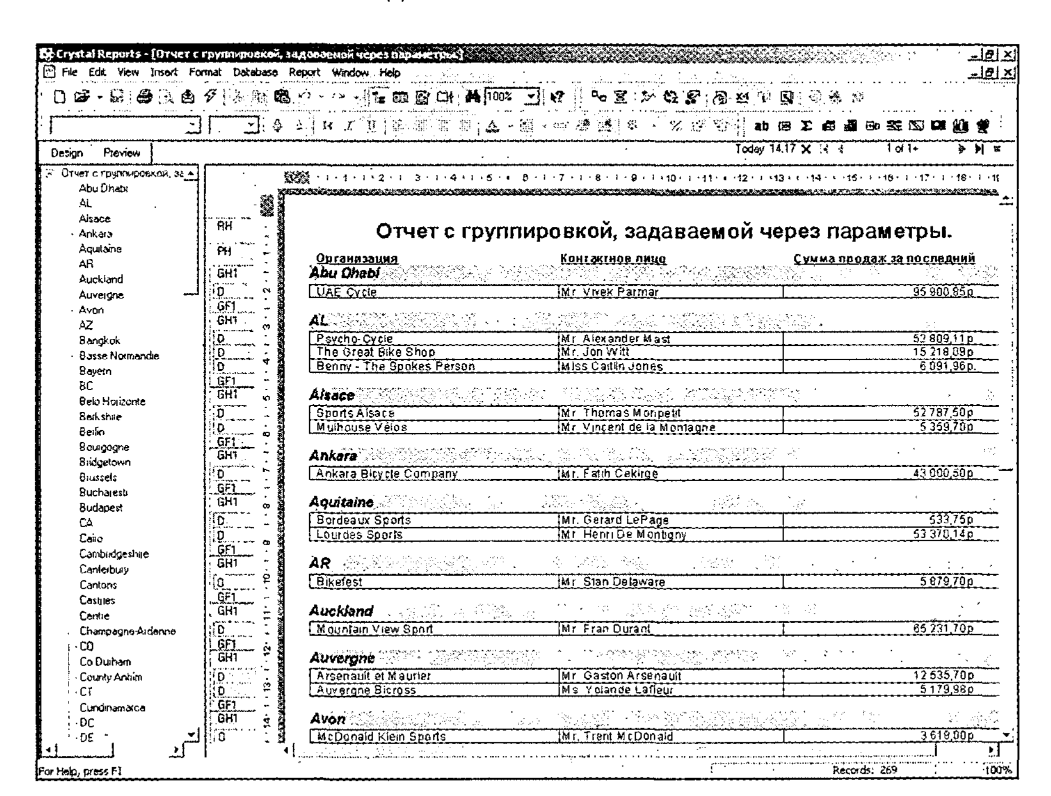


Рис. 9.13. Отчет с группированием на основании формулы, зависящей от параметра

Далее можно приступать к разработке приложения в Delphi или Visual Basic. При создании проекта не забывайте подключать файлы CRDelphi.pas или Clobal32.bas и CRwrap.bas. В проекте создайте экранную форму так, как это показано на рис. 9.14 и 9.15.

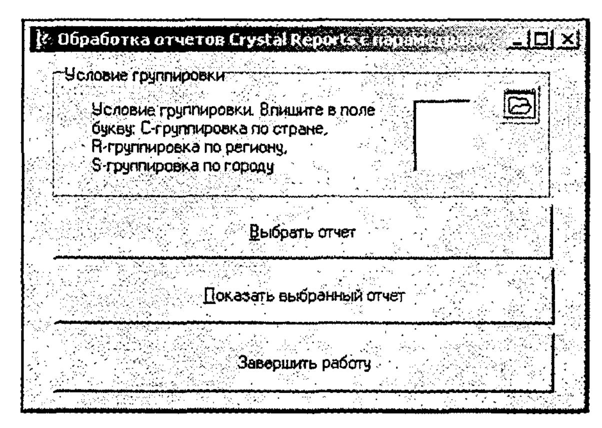


Рис. 9.14. Экранная форма для обработки отчета с параметром в среде Delphi

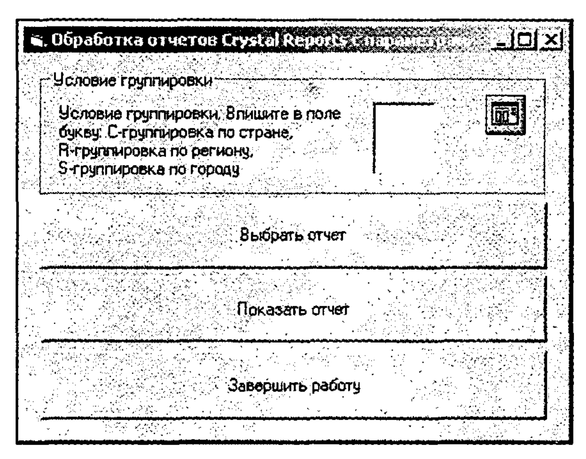


Рис. 9.15. Экранная форма для обработки отчета с параметром в среде Visual Basic

Все экранные элементы желательно назвать так, как в табл. 9.4.

Таблица 9.4. *Описание элементов экранного интерфейса приложения*

Тип объекта	Название	Текст, видимый на экране
Кнопка	btnSelectReport	Выбрать отчет
Кнопка	btnShowReport	Показать отчет

***Таблица 9.4** (окончание)*

Тип объекта	Название	Текст, видимый на экране
Кнопка	btnClose	Завершить работу
Поле	dfParamField	Используется для просмотра текущего значения и задания нового
Системное диалоговое окно, позволяющее выбрать отчет	ReportsOpenDialog	Настройки данного компонента выполняются по аналогии с предыдущими примерами

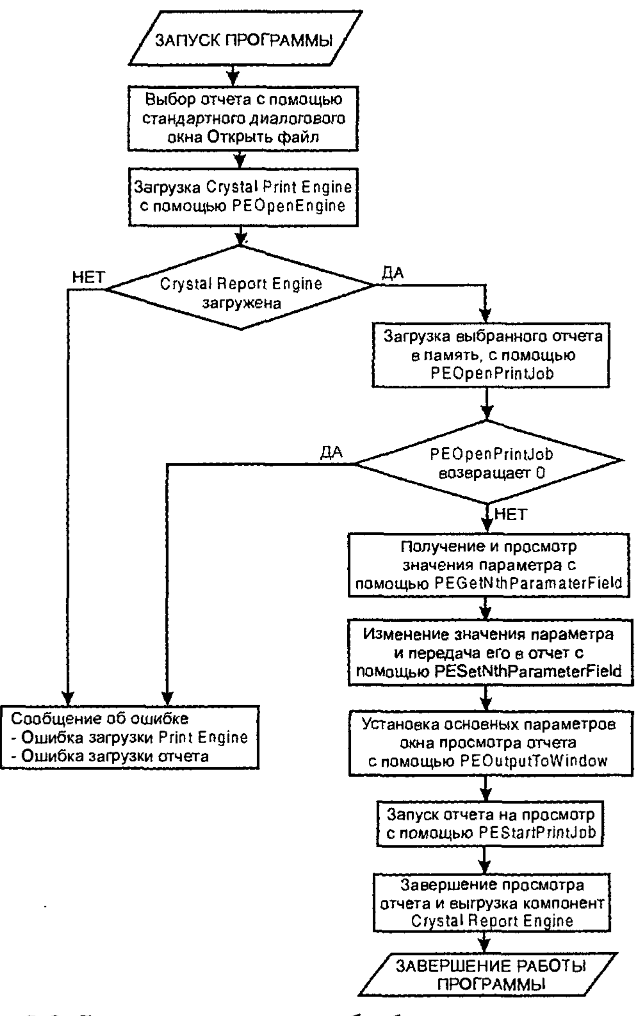


Схема 9.3. Блок-схема алгоритма обработки отчета с параметром

В предлагаемом примере желательно реализовать алгоритм, который показан на схеме 9.3.

Для реализации данного алгоритма необходимо сформировать процедуры так, как это показано в листингах 9.9 и 9.10. Обратите внимание на то, что в данных примерах используются дополнительные функции PEGetNthParameterField и PESetNthParameterField.

Листинг 9.9. Реализация процедур просмотра, изменения и передачи параметров в отчет в среде Delphi

```
unit CRW_API_Param;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, CRDelphi, StdCtrls;

type
  TfrmMainForm = class(TForm)
    GroupBox1: TGroupBox;
    dfParamField: TMemo;
    Label1: TLabel;
    btnSelectReport: TButton;
    ReportsOpenDialog: TOpenDialog;
    btnShowReport: TButton;
    btnClose: TButton;
    procedure btnSelectReportClick(Sender: TObject);
    procedure btnShowReportClick(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmMainForm: TfrmMainForm;
  nPrintJob: Word;
```

```
  vParamInfo:PEParameterFieldInfo;

implementation

{$R *.dfm}

procedure TfrmMainForm.btnSelectReportClick(Sender: TObject);
begin
   If ReportsOpenDialog.Execute Then
      Begin
        {Попытка запуска Crystal Reports Print Engine.
        Если запуск не удался, то выдается сообщение об
        ошибке.}
        If PEOpenEngine Then
          Begin
          {При корректном запуске Crystal Reports Print Engine
          выполняется попытка загрузить выбранный отчет.
          Если не удается загрузить отчет, то выдается сообщение
          об ошибке.}
          nPrintJob:=PEOpenPrintJob(PChar(ReportsOpenDialog.FileName));
          If nPrintJob=0 Then
             ShowMessage('Ошибка открытия отчета')
          Else
            Begin
             vParamInfo.structSize:= PE_SIZEOF_PARAMETER_FIELD_INFO;
             PEGetNthParameterField(nPrintJob,0,vParamInfo);
             dfParamField.Lines.Text:=vParamInfo.CurrentValue;
             dfParamField.SetFocus;
            End
          End
        End
        Else
           ShowMessage('Ошибка загрузки Crystal Report Engine');
end;

procedure TfrmMainForm.btnShowReportClick(Sender: TObject);
begin
   If nPrintJob<>0 Then
      Begin
```

```
        StrPCopy(vParamInfo.CurrentValue, dfParamField.Lines.Text)
        vParamInfo.structSize:= PE_SIZEOF_PARAMETER_FIELD_INFO;
        PESetNthParameterField(nPrintJob,0,vParamInfo);
        PEOutputToWindow(nPrintJob,
        PChar(ReportsOpenDialog.FileName), 0,0,0,0,
        WS_MAXIMIZE+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_SYSMENU, 0);
        PEStartPrintJob(nPrintJob,TRUE);
     End
end;

procedure TfrmMainForm.btnCloseClick(Sender: TObject);
begin
   If nPrintJob<>0 then
      Begin
         PEClosePrintJob(nPrintJob);
         PECloseEngine;
      End;
      Close;
end;

end.
```

Листинг 9.10. Реализация процедур просмотра, изменения и передачи параметров в отчет в среде Visual Basic

```
Dim nPrintJob As Integer
Dim vParamInfo As PEParameterFieldInfo
Dim result%

Private Sub btnSelectReport_Click()
    ReportsOpenDialog.ShowOpen
    If ReportsOpenDialog.fileName <> "" Then
   'Попытка запуска Crystal Reports Print Engine.
   'Если не удалось, то выдается сообщение об ошибке.
      If PEOpenEngine Then
        'При корректном запуске Crystal Reports Print Engine
        'выполняется попытка загрузить выбранный отчет.
        'Если не удается загрузить отчет, то выдается сообщение об ошибке.
        nPrintJob = PEOpenPrintJob(ReportsOpenDialog.fileName)
        If nPrintJob = 0 Then
```

```
            MsgBox ("Ошибка открытия отчета")
        Else
            'Извлечение текущего значения параметра
            vParamInfo.StructSize = PE_SIZEOF_PARAMETER_FIELD_INFO
            result% = PEGetNthParameterField(nPrintJob, 0, vParamInfo)
            dfParamField = vParamInfo.currentValue
        End If
      End If
    End If
End Sub

Private Sub btnShowReport_Click()
    If nPrintJob <> 0 Then
        'Подстановка нового значения параметра
        If dfParamField = "C" Then vParamInfo.currentValue = "C"
        End If
        If dfParamField = "R" Then vParamInfo.currentValue = "R"
        If dfParamField = "S" Then vParamInfo.currentValue = "S"
        'Передача нового значения параметра в отчет
        result% = PESetNthParameterField(nPrintJob, 0, vParamInfo)
        result% = PEOutputToWindow(nPrintJob,
        ReportsOpenDialog.fileName, 0, 0, 0, 0,
        WS_MAXIMIZE + WS_MINIMIZEBOX + WS_MAXIMIZEBOX + WS_SYSMENU, 0)
        result% = PEStartPrintJob(nPrintJob, True)
End Sub

Private Sub btnClose_Click()
    If nPrintJob <> 0 Then
        PEClosePrintJob (nPrintJob)
        PECloseEngine
    End If
    Unload Me
End Sub
```

В результате будет получено приложение, позволяющее работать с отчетом, который предлагалось создать в начале данного раздела.



В функциях PEGetNthParameterField и PESetNthParameterField в качестве второго параметра подставляется порядковый номер параметра. Нумерация параметров, всегда начинается с 0. Так как в отчете имеется всего лишь один параметр, его номер всегда 0.

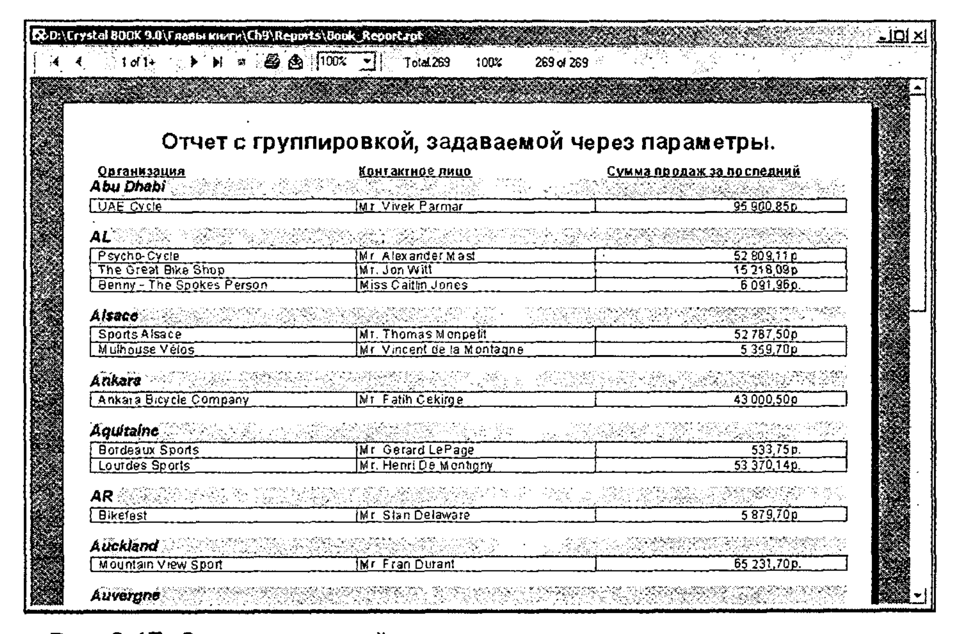
D:\Crystal BOOK 9.0\Главы книги\Ch9\Reports\Book_Report.rpt

1 of 1+ 100% Total:269 100%

Отчет с группировкой, задаваемой через параметры.

Организация	Контактное лицо	Сумма продаж за последний
Abu Dhabi		
UAE Cycle	Mr. Vivek Parmar	95 900,85р.
AL		
Psycho-Cycle	Mr. Alexander Mast	52 809,11р.
The Great Bike Shop	Mr. Jon Witt	15 218,09р.
Benny - The Spokes Person	Miss Caitlin Jones	6 091,96р.
Alsace		
Sports Alsace	Mr. Thomas Monpetit	52 787,50р.
Mulhouse Velos	Mr. Vincent de la Montagne	5 359,70р.
Ankara		
Ankara Bicycle Company	Mr. Fatih Cekirge	43 000,50р.
Aquitaine		
Bordeaux Sports	Mr. Gerard LePage	533,75р.
Lourdes Sports	Mr. Henri De Montigny	53 370,14р.
AR		
Bikefest	Mr. Stan Delaware	5 879,70р.
Auckland		
Mountain View Sport	Mr. Fran Durant	65 231,70р.
Auvergne		
Arsenault et Maurier	Mr. Gaston Arsenault	12 535,70р.
Auvergne Bicross	Ms. Yolande Lafleur	5 179,98р.

Рис. 9.16. Отчет, полученный при установке в качестве значения параметра латинского символа C



D:\Crystal BOOK 9.0\Главы книги\Ch9\Reports\Book_Report.rpt

1 of 1+ 100% Total:269 100% 269 of 269

Отчет с группировкой, задаваемой через параметры.

Организация	Контактное лицо	Сумма продаж за последний
Abu Dhabi		
UAE Cycle	Mr. Vivek Parmar	95 900,85р.
AL		
Psycho-Cycle	Mr. Alexander Mast	52 809,11р.
The Great Bike Shop	Mr. Jon Witt	15 218,09р.
Benny - The Spokes Person	Miss Caitlin Jones	6 091,96р.
Alsace		
Sports Alsace	Mr. Thomas Monpetit	52 787,50р.
Mulhouse Velos	Mr. Vincent de la Montagne	5 359,70р.
Ankara		
Ankara Bicycle Company	Mr. Fatih Cekirge	43 000,50р.
Aquitaine		
Bordeaux Sports	Mr. Gerard LePage	533,75р.
Lourdes Sports	Mr. Henri De Montigny	53 370,14р.
AR		
Bikefest	Mr. Stan Delaware	5 879,70р.
Auckland		
Mountain View Sport	Mr. Fran Durant	65 231,70р.
Auvergne		

Рис. 9.17. Отчет, полученный при установке в качестве значения параметра латинского символа R

D:\Crystal BOOK 9.0\Главы книги\Ch9\Reports\Book_Report.rpt

1 of 1+ 100% Total:269 100% 269 of 269

Отчет с группировкой, задаваемой через параметры.

Организация	Контактное лицо		Сумма продаж за последний
Abu Dhabi			
UAE Cycle	Mr. Vivek Parmar		95 900,85р.
Acapulco			
Tiempo Libre Monterrey	Mr. Jose Garcia		8 819,55р.
Amiens			
Vélos Emile	Mr. Emile Laroche		4 366,50р.
Amsterdam			
Canal City Cycle	Mr. Sven Vandervoort		100 000,00р.
Ankara			
Ankara Bicycle Company	Mr. Fatih Cekirge		43 000,50р.
Arras			
Bicyclette du Nord	Ms. Elise Marchand		8 819,55р.
Ashford			
Sunrise Cycle	Mrs. Josephine McGillan		1 690,05р.
Athens			
Athens Bicycle Co	Mr. Andres Arnadis		11 100,90р.
Atlanta			
Tony's Better Bikes	Mr. Anthony Weatherhead		2 735,25р.

Рис. 9.18. Отчет, полученный при установке в качестве значения параметра латинского символа S

При изменении значения параметра в поле полученного приложения отчет будет выведен на экран так, как это показано на рис. 9.16, 9.17 и 9.18.



Предложенный пример позволяет оценить возможности использования функций Crystal Report API при работе с отчетами, содержащими параметры. Однако если воспользоваться функциями в полном объеме, можно работать не только с заранее известными отчетами, но и выполнять обработку отчетов с параметрами любой сложности и с другими дополнительными элементами. Так же, используя функции Crystal Report API, можно работать с условия фильтрации данных и редактировать их, если это необходимо.

9.2.4. Работа с различными источниками данных. Использование Crystal Report API

При работе с отчетом часто возникает необходимость переопределить источник данных, который будет использоваться для формирования результата. Если пользоваться непосредственно Crystal Reports 9, то можно переопределить источник данных так, как это описано в *главе 7*. При разработке

приложения, вызывающего отчет, хотелось бы иметь механизм, позволяющий выполнять переключение от одного источника к другому, настраивая параметры непосредственно в программе. Для выполнения такого рода действий существует большой набор функций, позволяющих получить информацию об источнике данных, используемом в отчете по умолчанию, и изменить источник и требуемые параметры. К этим функциям относятся следующие:

- **PEGetNthTableType** — извлекает информацию о библиотеке, используемой для подключения к базе данных;
- **PEGetNthTableLogOnInfo** — извлекает информацию об имени сервера базы данных, базе данных, имени пользователя и пароле, пароль всегда возвращается пустой строкой;

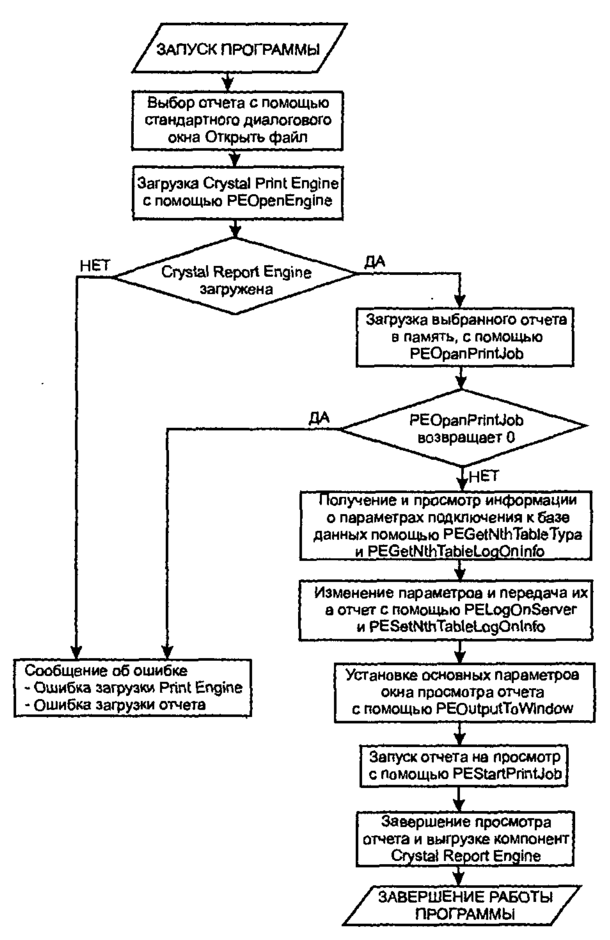


Схема 9.4. Блок-схема алгоритма, позволяющего менять параметры подключения отчета к базе данных из программы

- **PELogOnServer** — передает новые параметры подключения к базе данных в отчет;
- **PESetNthTableLogOnInfo** — определяет список таблиц, к которым необходимо применить новые параметры подключения.

Рассмотрим простейший пример работы с этими функциями и реализуем приложение, которое работает по алгоритму, представленному на схеме 9.4.



Пример, описанный ниже, предполагает использование различных источников данных. При наличии расхождений в структуре данных для разных источников этот пример работать не будет, так как для корректировки расхождений потребуется использовать еще ряд функций, позволяющих связать новые таблицы и поля с имеющимся шаблоном отчета.

Используя правила, изложенные в предыдущих разделах, создайте новый проект в Delphi или Visual Basic. Реализуйте экранную форму, которая показана на рис. 9.19 и 9.20.

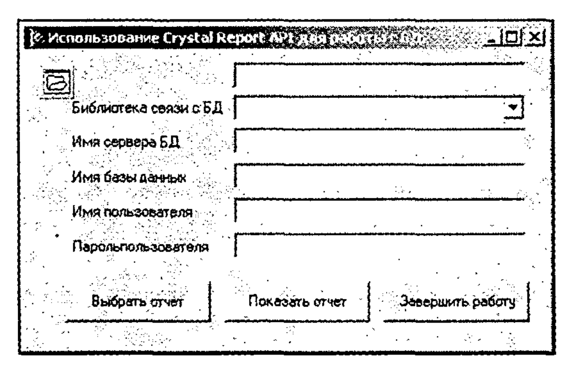


Рис. 9.19. Экранная форма с элементами контроля для просмотра и изменения параметров подключения отчета к базе данных в среде Delphi

Рис. 9.20. Экранная форма с элементами контроля для просмотра и изменения параметров подключения отчета к базе данных в среде Visual Basic

Для того чтобы воспользоваться представленными ниже листингами, желательно объекты в экранной форме определить так, как это описано в табл. 9.5.

Таблица 9.5. Список компонентов, включенных в экранную форму

Тип объекта	Название	Текст на экране
Кнопка	btnSelectReport	Выбрать отчет
Кнопка	btnShowReport	Показать отчет
Кнопка	btnClose	Завершить работу
Системное диалоговое окно, позволяющее выбрать отчет	ReportsOpenDialog	Настройки данного компонента выполняются по аналогии с предыдущими примерами
Поле	dfLibrary	Описание библиотеки, используемой в отчете
Поле	dfServerName	Имя сервера базы данных
Поле	dfDatabase	Имя базы данных
Поле	dfUserName	Имя пользователя
Поле	dfPassword	Имя базы данных
ComboBox	cmbDatabaseLibrary	Имя библиотеки, используемой в отчете. Выбирается из списка

У компоненты ComboBox заранее необходимо определить список значений:

- **crdb_p2bbde.dll** — Borland Database Engine
- **crdb_dao.dll** — DAO data sources (Access)
- **crdb_p2bbde.dll** — Paradox
- **crdb_p2bxbse.dll** — dBASE, FoxPro, Clipper
- **crdb_p2bbtrv.dll** — Btrieve
- **crdb_p2sdb2.dll** — DB2/2
- **crdb_odbc.dll** — ODBC. See Remarks below
- **crdb_oracle.dll** — Oracle
- **crdb_p2ssyb10.dll** — Sybase 10/11

Для этого в проекте Delphi необходимо, установив фокус на компоненту, перейти в окно Object Inspector и сформировать указанный список для свойства Items, добавив в начало пустую строку. Кроме этого, необходимо установить значение свойства ItemIndex равным 0. Это делается затем, чтобы в момент запуска приложения выводилась пустая строчка в ComboBox.

В проекте Visual Basic для такого же компонента требуется, перейдя в редактор свойств Properties, заполнить свойство List рекомендуемым списком имен, добавив в начало списка пустую строку. Свойство Index должно быть равно 0.

Настроив свойства компоненты OpenDialog для Delphi и CommonDialog для Visual Basic, так как это предложено в ранее описанных примерах, можно приступить к написанию программы. Текст программы показан в листингах 9.11 и 9.12.

Листинг 9.11. Текст программы в Delphi, реализующей алгоритм схемы 9.4

```
unit frmMainForm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, CRDelphi, StdCtrls;

type
  TMain_Form = class(TForm)
    dfLibrary: TEdit;
    dfServerName: TEdit;
    dfDatabase: TEdit;
    dfUserName: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    btnSelectReport: TButton;
    btnShowReport: TButton;
    btnClose: TButton;
    ReportsOpenDialog: TOpenDialog;
    dfPassword: TEdit;
    cmbDatabaseLibrary: TComboBox;
    procedure btnSelectReportClick(Sender: TObject);
    procedure cmbDatabaseLibraryClick(Sender: TObject);
    procedure btnShowReportClick(Sender: TObject);
```

```
    procedure btnCloseClick(Sender: TObject};
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Main_Form: TMain_Form;
  nPrintJob: Word;
  bEngine: Boolean;
  vLogOnInfo: PELogOnInfo;
  vLibraryName: PETableType;

implementation

{$R *.dfm}

{Процедура выбора отчета и загрузки параметров, связанных с базой данных.
Обрабатывается при нажатии иконки "Выбрать отчет"}
procedure TMain_Form.btnSelectReportClick(Sender: TObject);
begin
   If ReportsOpenDialog.Execute Then
      Begin
        {Попытка запуска Crystal Reports Print Engine.
        Если запуск не удался, то выдается сообщение об
        ошибке.}
        bEngine := PEOpenEngine;
        If bEngine Then
          Begin
          {При корректном запуске Crystal Reports Print Engine
          выполняется попытка загрузить выбранный отчет.
          Если не удается загрузить отчет, то выдается сообщение
          об ошибке.}
          nPrintJob:=PEOpenPrintJob(PChar(ReportsOpenDialog.FileName));
          If nPrintJob=0 Then
             ShowMessage('Ошибка открытия отчета')
          Else
            Begin
```

```
                {Получение информации о библиотеке,
                 используемой для подключения к базе данных}
                vLibraryName.StructSize := PE_SIZEOF_TABLE_TYPE;
                PEGetNthTableType(nPrintJob,0,vLibraryName);
                cmbDatabaseLibrary.Text := vLibraryName.DLLName;
                cmbDatabaseLibraryClick(Self);
                {Получение информации о сервере,
                 базе данных, имени пользователя и пароле}
                vLogOnInfo.StructSize := PE_SIZEOF_LOGON_INFO;
                PEGetNthTableLogOnInfo(nPrintJob,0,vLogOnInfo};
                dfServerName.Text := vLogOnInfo.ServerName;
                dfDatabase.Text := vLogOnInfo.DatabaseName;
                dfUserName.Text := vLogOnInfo.UserId;
                dfPassword.Text := vLogOnInfo.Password;
            End
          End
        End
        Else
           ShowMessage('Ошибка загрузки Crystal Report Engine')
end;

{Процедура изменения библиотеки связи с базой данных. Применяется при
выбоpе значения из ComboBox}
procedure TMain_Form.cmbDatabaseLibraryClick(Sender: TObject);
var
  i: Integer;
begin
        i := 0;
        while (i < cmbDatabaseLibrary.Items.Count) and
        (cmbDatabaseLibrary.Items[i] <> cmbDatabaseLibrary.Text) do
        inc(i);
        Case i of
                0: dfLibrary.Text:='База данных не выбрана';
                1: dfLibrary.Text:='Borland Database Engine';
                2: dfLibrary.Text:='DAO data sources (Access}';
                3: dfLibrary.Text:='Paradox';
                4: dfLibrary.Text:='dBASE, FoxPro, Clipper';
                5: dfLibrary.Text:='Btrieve';
                6: dfLibrary.Text:='DB2/2';
```

```
            7: dfLibrary.Text:='ODBC';
            8: dfLibrary.Text:='Oracle';
            9: dfLibrary.Text:='Sybase 10/11';
      End;
end;

{Процедура запуска отчета с новыми параметрами подключения к базе данных.
Применяется при нажатии кнопки "Показать отчет"}
procedure TMain_Form.btnShowReportClick(Sender: TObject);
begin
        {Изменение информации о параметрах подключения
         к базе данных}
        vLogOnInfo.StructSize := PE_SIZEOF_LOGON_INFO;
        StrPCopy(vLogOnInfo.ServerName, dfServerName.Text);
        StrPCopy(vLogOnInfo.DatabaseName, dfDatabase.Text);
        StrPCopy(vLogOnInfo.UserID, dfUserName.Text);
        StrPCopy(vLogOnInfo.Password, dfPassword.Text);
        {Передача параметров подключения к базе данных в отчет}
        If not PELogOnServer(PChar(cmbDatabaseLibrary.Text), vLogOnInfo)
        then
           ShowMessage('Ошибка передачи параметров соединения
                        с базой данных');
        {Привязка таблиц в отчете к указанным параметрам.
         Вызов данной процедуры обязателен.}
        If not PESetNthTableLogOnInfo(nPrintJob,0,vLogOnInfo,TRUE) then
          ShowMessage('Ошибка передачи параметров соединения
                       с базой данных');
        PEOutputToWindow(nPrintJob,
        PChar(ReportsOpenDialog.FileName), 0,0,0,0,
        WS_MAXIMIZE+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_SYSMENU, 0);
        PEStartPrintJob(nPrintJob,TRUE);
end;

{Процедура завершения работы отчета}
procedure TMain_Form.btnCloseClick(Sender: TObject);
begin
        If (bEngine=TRUE) Then
        Begin
            If nPrintJob<>0 Then
                PEClosePrintJob(nPrintJob);
```

```
            PECloseEngine;
        End;
        Close;
end;

end.
```

Листинг 9.12. Текст программы в Visual Basic, реализующей алгоритм схемы 9.4

```
Dim result%
Dim nPrintJob As Integer
Dim bEngine As Boolean
Dim vLogOnInfo As PELogOnInfo
Dim vLibraryName As PETableType

'Процедура выбора отчета и загрузки параметров, связанных с базой данных.
'Применяется при нажатии иконки "Выбрать отчет".
Private Sub btnSelectReport_Click()
    ReportsOpenDialog.ShowOpen
    If ReportsOpenDialog.fileName <> "" Then
   'Попытка запуска Crystal Reports Print Engine.
   'Если не удалось, то выдается сообщение об ошибке.
      bEngine = PEOpenEngine
      If bEngine Then
       'При корректном запуске Crystal Reports Print Engine
       'выполняется попытка загрузить выбранный отчет.
       'Если не удается загрузить отчет, то выдается сообщение об ошибке.
        nPrintJob = PEOpenPrintJob(ReportsOpenDialog.fileName)
        If nPrintJob = 0 Then
            MsgBox ("Ошибка открытия отчета")
        Else
            'Получение информации о библиотеке,
            'используемой для подключения к базе данных
            vLibraryName.StructSize = PE_SIZEOF_TABLE_TYPE
            result% = PEGetNthTableType(nPrintJob, 0, vLibraryName)
            cmbDatabaseLibrary.Item(0) = vLibraryName.DLLName
            cmbDatabaseLibrary_Click (0)
            'Получение информации о сервере,
            'базе данных, имени пользователя и пароле
            vLogOnInfo.StructSize = PE_SIZEOF_LOGON_INFO
```

```
            result% = PEGetNthTableLogOnInfo(nPrintJob, 0, vLogOnInfo)
            dfServerName = vLogOnInfo.ServerName
            dfDatabase = vLogOnInfo.DatabaseName
            dfUserName = vLogOnInfo.UserID
            dfPassword = vLogOnInfo.Password

        End If
      Else
        MsgBox ("Ошибка запуска Print Engine")
      End If
    End If

End Sub

'Процедура запуска отчета с новыми параметрами подключения к базе данных.
'Применяется при нажатии кнопки "Показать отчет".
Private Sub btnShowReport_Click()
    'Изменение информации о параметрах подключения
    'к базе данных
    vLogOnInfo.StructSize = PE_SIZEOF_LOGON_INFO
    vLogOnInfo.ServerName = dfServerName + String(128 -
    Len(dfServerName), Chr$(0))
    vLogOnInfo.DatabaseName = dfDatabase + String(128 -
    Len(dfDatabase), Chr$(0))
    vLogOnInfo.UserID = dfUserName + String(128 -
    Len(dfUserName), Chr$(0))
    vLogOnInfo.Password = dfPassword + String(128 -
    Len(dfPassword), Chr$(0))
    'Передача параметров подключения к базе данных в отчет
    result% = PELogOnServer(cmbDatabaseLibrary.Item(0), vLogOnInfo)
    If result% = 0 Then
        MsgBox ("Ошибка передачи параметров соединения с базой данных")
    End If
    'Привязка таблиц в отчете к указанным параметрам.
    'Вызов данной процедуры обязателен.
    result% = PESetNthTableLogOnInfo(nPrintJob, 0, vLogOnInfo, True)
    If result% = 0 Then
        MsgBox ("Ошибка передачи параметров соединения с базой данных")
    End If
```

```
    result% = PEOutputToWindow(nPrintJob,
           ReportsOpenDialog.fileName, 0, 0, 0, 0,
           WS_MAXIMIZE + WS_MINIMIZEBOX + WS_MAXIMIZEBOX + WS_SYSMENU, 0)
    result% = PEStartPrintJob(nPrintJob, True)
End Sub

'Процедура изменения библиотеки связи с базой данных.
'Применяется при выборе значения из ComboBox
Private Sub cmbDatabaseLibrary_Click(Index As Integer)
       Select Case cmbDatabaseLibrary.Item(0)
        Case "":     dfLibrary = "База данных не выбрана"
        Case "crdb_p2bbde.dll":     dfLibrary = "Borland Database Engine"
        Case "crdb_dao.dll":     dfLibrary = "DAO data sources (Access)"
        Case "crdb_p2bbde.dll":     dfLibrary = "Paradox"
        Case "crdb_p2bxbse.dll":     dfLibrary = "dBASE, FoxPro, Clipper"
        Case "crdb_p2bbtrv.dll":     dfLibrary = "Btrieve"
        Case "crdb_p2sdb2.dll":     dfLibrary = "DB2/2"
        Case "crdb_odbc.dll":     dfLibrary = "ODBC"
        Case "crdb_oracle.dll":     dfLibrary = "Oracle"
        Case "crdb_p2ssyb10.dll":     dfLibrary = "Sybase 10/11"
       End Select

End Sub

'Процедура завершения работы отчета.
Private Sub btnClose_Click()
    If bEngine Then
        If nPrintJob <> 0 Then
            PEClosePrintJob (nPrintJob)
        End If
        PECloseEngine
    End If
    Unload Me
End Sub
```

После отладки и компиляции данного программного кода можно запустить приложение и, выбрав какой-либо отчет, увидеть информацию о параметрах связи отчета с базой данных так, как это показано на рис. 9.21 и 9.22.

Рис. 9.21. Программа Delphi для работы с параметрами подключения к базе данных

Рис. 9.22. Программа Visual Basic для работы с параметрами подключения к базе данных

Изменив параметры в полях экранной формы, можно вывести выбранный отчет на просмотр, нажав кнопки **Показать отчет**. Как уже было сказано ранее, структура источника данных, для которого задаются новые параметры, должна совпадать со структурой того источника, на основании которого строился отчет. Иначе на экран ничего не будет выведено. Для того чтобы усложнить данные примеры и получить больше возможностей для работы с параметрами подключения отчета к базе данных, необходимо пользоваться полным спектром функций, описанных в *Приложении 1*.

9.3. Использование компонент ActiveX

Помимо возможности, предоставленной Crystal Report Print Engine API, существуют другие подходы к реализации приложений, работающих с отчетами Crystal Report 9. Одним из таких механизмов является возможность использования ActiveX компонент. В Crystal Reports версии 8.5 и младше присутствовал компонент, который можно было использовать в любых средствах разработки приложений, позволяющих работать с ActiveX компонентами. Этот компонент использует функции из библиотеки Crystal Report Print Engine API. Но, к сожалению, его функциональное наполнение не менялось со времен Crystal Reports версии 5.X. Поэтому большое количество функций, добавленных в Crystal Reports старших версий, не работают в данной компоненте. Начиная с Crystal Reports 9.X данный компонент более не входит в поставку. Вместо него предлагается использовать Crystal Report Designer Component. Данные компоненты позволяют не только выполнять интеграцию отчетов Crystal Reports 9 в приложения, но и создавать свою собственную оболочку, дающую возможность создавать новые отчеты.

Примечание

Приобретая пакет Crystal Reports 9.0 Developer или Advanced, вы получаете возможность выполнять разработку приложений с применением компонентов Crystal Report Designer Components. Однако использовать разработанное приложение можно будет только на том компьютере, где выполнялась разработка. Для установки приложения на другие компьютеры потребуется дополнительная лицензия, которая меняется в зависимости от потребностей. За уточнениями, относящимися к лицензированию, необходимо обращаться непосредственно в компанию Crystal Decisions.

Использовать Crystal Report Designer Components можно только при разработке приложений с помощью Visual Basic или Visual C++.

Для того чтобы понять, как использовать данный компонент, создадим приложение в Visual Basic. Чтобы обеспечить использование в приложении компонент Crystal Report Designer Components, необходимо выполнить следующие действия.

1. В Visual Basic создается стандартный проект.
2. Выбирается команда меню **Project | Add Crystal Reports 9**.
3. В появившемся диалоговом окне, которое показано на рис. 9.23, выбирается вариант работы с отчетом по аналогии с Crystal Reports 9:
   - Using The Report Expert — использовать эксперт;
   - As a Blank Report — использовать настраиваемый самостоятельно бланк отчета;
   - From an Existing Report — использовать существующий отчет.

4. Для ознакомления выбирается вариант From an Existing Report. При этом, после выбора отчета, появляется диалоговое окно, показанное на рис. 9.24. В этом окне предлагается установить параметры изменения проекта. Первоначально желательно в этом диалоге ничего не менять.

5. Проект изменится автоматически. В него будут добавлены компоненты, позволяющие отредактировать отчет так же, как это делается в Crystal Reports 9, и новая форма, содержащая компоненту CRViewer. Эти изменения в проекте показаны на рис. 9.25 и 9.26.

6. Далее требуется изменить название добавленной в проект формы Form2 на другое, предположим, frmReportViewer, и описать метод, относящийся к загрузке формы так, как это показано в листинге 9.13.



При использовании предложенного алгоритма, весь программный код будет сформирован автоматически. После его проверки можно запускать получившееся приложение.

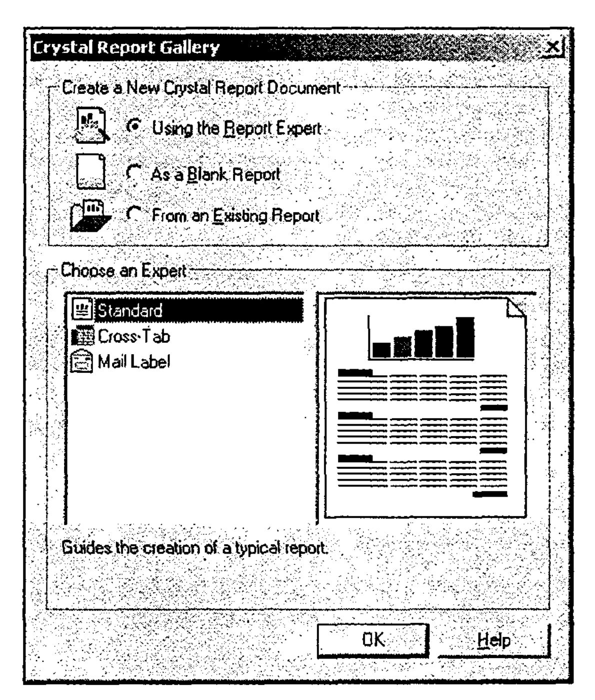


Рис. 9.23. Окно **Crystal Report Gallery**. Использование **Crystal Report Designer Components**

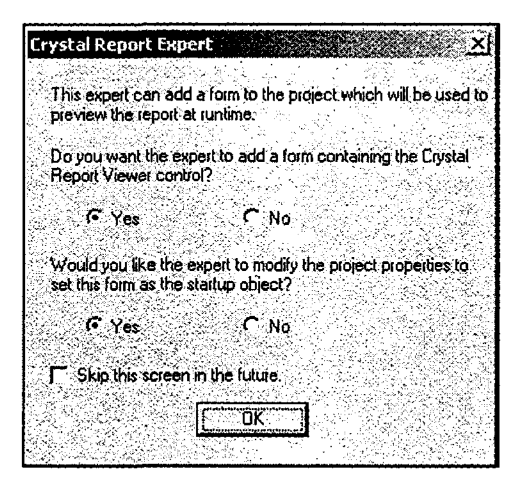


Рис. 9.24. Окно настройки параметров изменения проекта. Использование **Crystal Report Designer Components**

Рис. 9.25. Дополнение, внесенное в проект. Среда редактирования отчета

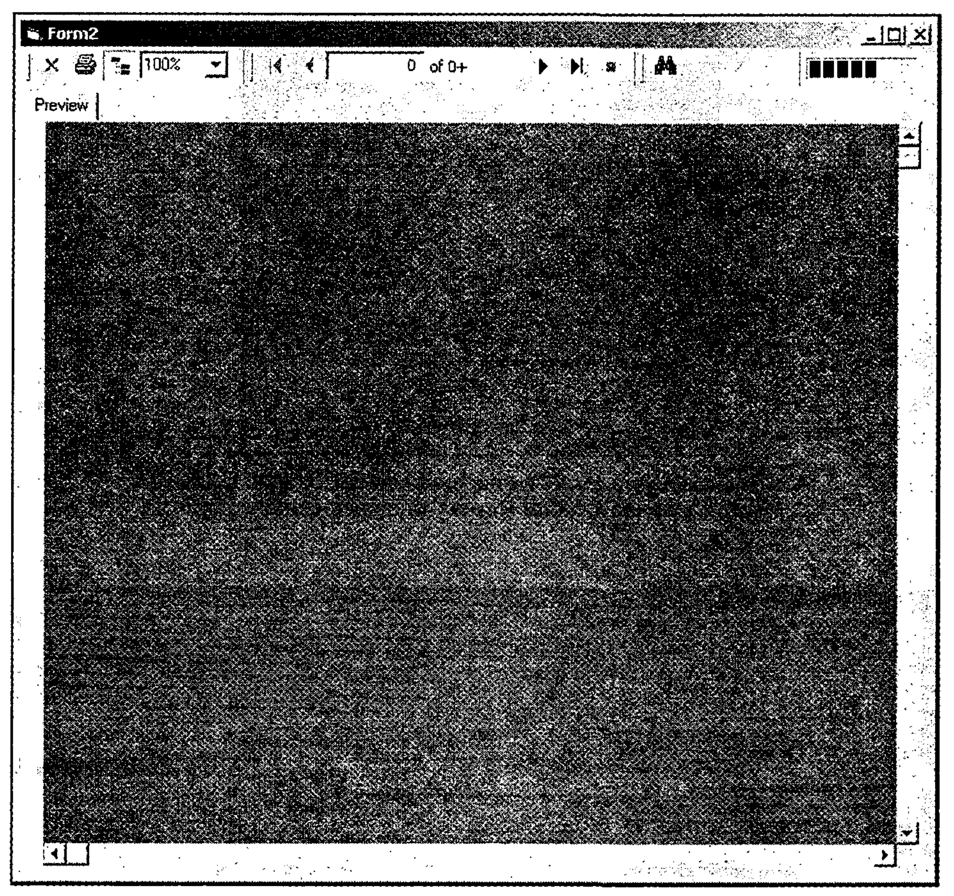


Рис. 9.26. Экранная форма, содержащая компоненту **Crystal Viewer**

Листинг 9.13. Проект, относящийся к загрузке экранной формы с компонентом просмотра отчета. Данный текст формируется автоматически

```
Private Sub Form_Load()
    Screen.MousePointer = vbHourglass
    CRViewer91.ReportSource = Report
    CRViewer91.ViewReport
    Screen.MousePointer = vbDefault
End Sub
```

В результате после запуска получившегося приложения на экран выводится экранная форма с окном просмотра отчета (рис. 9.27).

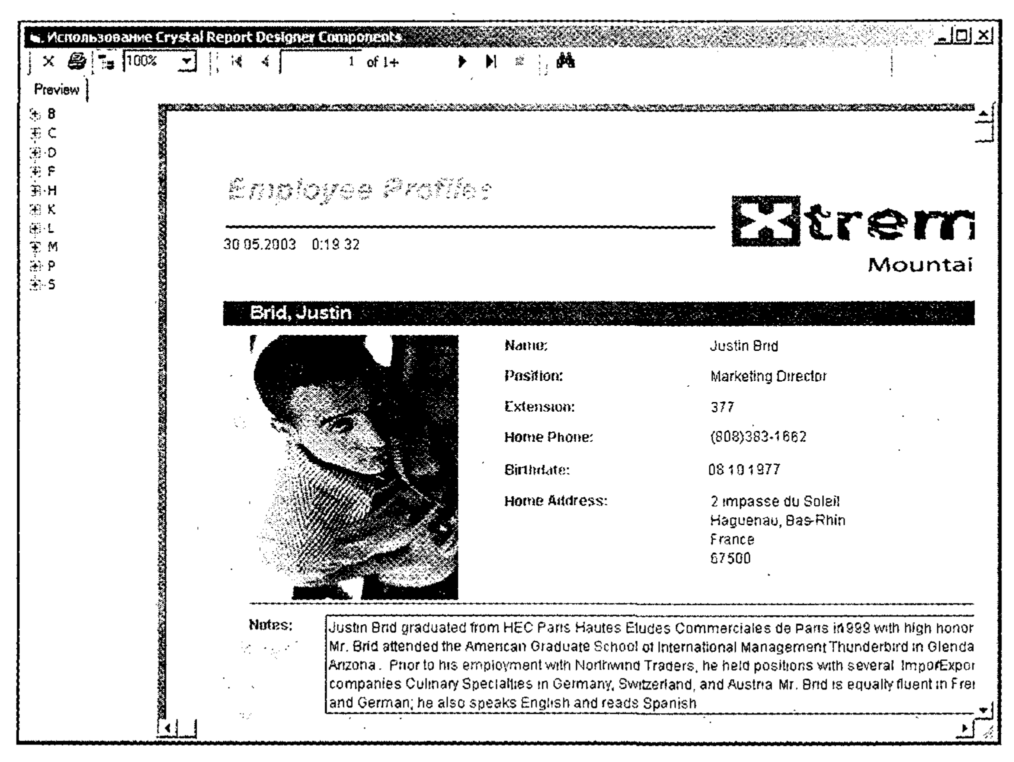


Рис. 9.27. Результат использования **Crystal Report Designer Components**

Предложенный механизм использования этих компонентов является самым простым. Как было сказано ранее, существует большое количество способов использования данных компонентов. Но даже для такого варианта приложения есть еще много методов, позволяющих управлять окном просмотра:

- **DisplayBackgroundEdge** — механизм отображения или скрытия края страницы;
- **DisplayBorder** — показывать механизм отображения или скрытия рамок;
- **DisplayGroupTree** — механизм отображения или скрытия дерева групп;
- **DisplayTabs** — механизм отображения или скрытия вкладок;
- **DisplayToolbar** — механизм отображения или скрытия панели инструментов;
- **EnableAnimationCtrl** — механизм, обеспечивающий доступ к элементам анимации;
- **EnableCloseButton** — механизм, обеспечивающий доступ к кнопке закрытия отчета;
- **EnableDrillDown** — механизм, разрешающий выполнять специальную выборку в отчете;

- **EnableExportButton** — механизм, обеспечивающий доступ к кнопке экспорта;
- **EnableGroupTree** — механизм, обеспечивающий доступ к кнопке показа дерева групп;
- **EnableHelpButton** — механизм, обеспечивающий доступ к кнопке помощи;
- **EnableNavigationControls** — механизм, обеспечивающий доступ к кнопкам навигации;
- **EnablePopupMenu** — механизм, обеспечивающий доступ к выпадающему меню;
- **EnablePrintButton** — механизм, обеспечивающий доступ к кнопке печати;
- **EnableProgressControl** — механизм, обеспечивающий отображение индикатора прогресса;
- **EnableRefreshButton** — механизм, обеспечивающий доступ к кнопке обновления;
- **EnableSearchControl** — механизм, обеспечивающий доступ к элементам поиска;
- **EnableSearchExpertButton** — механизм, обеспечивающий доступ к кнопке вызова эксперта поиска;
- **EnableSelectExpertButton** — механизм, обеспечивающий доступ к кнопке вызова диалогового окна настройки фильтрации данных в отчете;
- **EnableStopButton** — механизм, обеспечивающий доступ к кнопке прекращения загрузки данных;
- **EnableToolbar** — механизм, обеспечивающий доступ к кнопке в панели инструментов;
- **EnableZoomControl** — механизм, обеспечивающий доступ к элементам управления размерами отчета.

О более развитых возможностях компонентов Crystal Report Designer Components можно узнать из документации Crystal Developers Guide, которая поставляется совместно в Crystal Reports 9.0 Developer или Advanced.

9.4. Использование компонентов для Delphi

Для удобства разработчиков, использующих Delphi в качестве среды разработки приложений, работающих с отчетами Crystal Reports 9, предусмотрен компонент, который поставляется с Crystal Reports 9.0 Developer или Advanced. В момент установки Crystal Reports 9 можно выбрать требуемый компонент, он будет скопирован в каталог ..\Program Files\Crystal Decisions\Crystal Reports 9\Samples\En\Code\Delphi\cr85vcl.exe. При запуске данного приложения будет выполнен процесс установки пакетов, требуемых для работы данного компонента в Delphi.



Как видно из названия файла, данный пакет не изменял свои свойства с Crystal Reports версии 8.5.

В момент установки необходимо следовать рекомендациям программы установки. Программа автоматически определит версию Delphi и предложит каталог для записи нужных пакетов. При необходимости некоторые параметры можно изменить. По завершении установки в палитре инструментов Delphi, на вкладке Data Access, появятся дополнительные компоненты и . Расшифрованное наименование данных компонентов:

- **CRPE** — Crystal Report Print Engine;
- **CRPE DS** — Crystal Report Print Engine Data Source.

Названия указывают на то, что данные компоненты используют для своей работы функции Crystal Report Print Engine API, некоторые из которых описаны в данной главе в *разд. 9.2*, а полный список этих функций представлен в *Приложении 1* и *Приложении 2.*

Чтобы убедиться в работоспособности данного компонента, имеет смысл выполнить простейший тест. Открыв новый проект, создайте экранную форму так, как это показано на рис. 9.28.

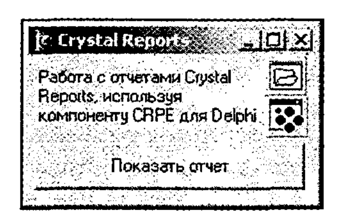


Рис. 9.28. Экранная форма, содержащая компонент для работы с отчетами **Crystal Reports**

Настройте компонент OpenDialog так, как это предложено в ранее рассмотренных примерах. Переименуйте компонент CRPE1 в CrystalRptComponent. Установите у кнопки свойство Name — btnShowReport и напишите простейший программный код, показанный в листинге 9.14.

Листинг 9.14. Программа, позволяющая запустить на просмотр выбранный отчет. Использование компонента CRPE

```
procedure TForm1.btnShowReportClick(Sender: TObject);
begin
   If ReportsOpenDialog.Execute Then
```

```
    Begin
       CrystalRptComponent.ReportName := ReportsOpenDialog.FileName;
       CrystalRptComponent.Execute;
    End
end;
```

Запустив данную программу на выполнение и выбрав любой доступный отчет, вы увидите его на экране в окне просмотра. Приведенный пример показывает, что использование компонента в Delphi намного упрощает разработку приложений, применяющих отчеты, созданные в Crystal Reports 9. Так как компоненты для Delphi созданы с использованием функций Crystal Report Print Engine API, разработчики получают полный набор возможностей по работе с отчетами.

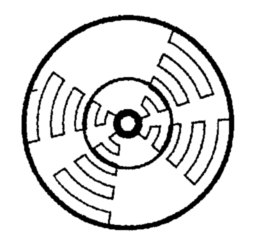
9.5. Дополнительные варианты интеграции отчетов Crystal Reports в приложения

Помимо описанных в предыдущих разделах возможностей интеграции отчетов Crystal Reports 9 в приложения, существует новый механизм, предназначенный для использования отчетов из Java-приложений. По аналогии с компонентами Crystal RDC разработана компонента Crystal Report Viewer Bean. Этот компонент, так же как компоненты Crystal RDC, можно добавить в приложение, создаваемое с помощью средств визуального программирования Java. К сожалению, он работает некорректно в наиболее распространенных средствах Java-программирования, таких как Borland JBuilder, Oracle JDeveloper и т. п. Для его работы необходимо установить пакет Java 2 SDK 1.1, который можно бесплатно загрузить с Web-сайта фирмы-изготовителя Sun Microsystems. Для работы данного компонента необходимо наличие Web-сервера, чтобы обеспечить обращение к отчету так же, как это делается при обращении к любой Web-странице. Результатом разработки приложения с использованием данного компонента является Java-апплет, на который устанавливается ссылка из HTML-страницы. Описание и пример интеграции отчетов Crystal Reports в Java-апплеты представлены в документации разработчика Crystal Reports Developers Guide.

Заключение

Материал данной главы не претендует на полноту и идеальность. Однако автор надеется, что в предоставленном материале содержится информация, которая позволит опытным разработчикам быстро разобраться с имеющимися возможностями и использовать их в своей работе, а новички получат дополнительные знания, которые пригодятся им в дальнейшем.

Приложение 1



Список функций Crystal Report Print Engine API

В *Приложении 1* и *Приложении 2* представлен список функций и определяемых пользователем типов данных, которые иногда именуются как записи или структуры. Все описания даны применительно к языку Object Pascal системы Delphi. Однако, используя предложенную информацию, можно реализовать работу с функциями в любом средстве разработки, поддерживающем использование функций API.

Замечание

В документации, которая поставляется совместно с Crystal Reports 9 и относится к разделу Crystal Legacy SDK, можно найти описание функций и типов для Microsoft Visual Basic и C++. В этой документации присутствует также большое количество примеров, которые будут полезны разработчикам.

Функция `PEAddParameterCurrentRange` — добавляет значения, соответствующие требуемому диапазону, в указанное поле-параметр отчета.

Замечание

В данный момент поля такого типа не поддерживаются в Web Viewer.

```
procedure PEAddParameterCurrentRange(
    printJob: smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    var rangeStart: PEValueInfo;
    var rangeEnd: PEValueInfo;
    rangeInfo: smallint;
    ): BOOL stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который необходимо добавить значение параметра диапазона
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
rangeStart	Запись PEValueInfo, которая содержит значение нижней границы диапазона
rangeEnd	Запись PEValueInfo, которая содержит значение верхней границы диапазона
rangeInfo	Значение, которое указывает на необходимость учета верхней или нижней границы. Можно использовать комбинацию констант:
	Константа PE_RI_INCLUDEUPPERBOUND
	Значение 1
	Константа PE_RI_INCLUDELOWERBOUND
	Значение 2
	Константа PE_RI_NOUPPERBOUND
	Значение 4
	Константа PE_RI_NOLOWERBOUND
	Значение 8

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEAddParameterCurrentValue** — добавляет значение параметра в указанное поле параметр отчета.

Замечание

В данный момент поля такого типа не поддерживаются в Web Viewer.

```
procedure PEAddParameterCurrentValue (
    printJob: smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    var currentValue: PEValueInfo
    ): BOOL stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который необходимо добавить значение параметра
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
currentValue	Запись PEValueInfo, которая содержит добавляемое в поле значение параметра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEAddParameterDefaultValue** — добавляет значение в группу значений по умолчанию для указанного параметра в отчете.

```
procedure PEAddParameterDefaultValue (
    printJob: smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    var valueInfo: PEValueInfo
    ): BOOL stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который необходимо добавить значение параметра по умолчанию
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
valueInfo	Запись PEValueInfo, которая содержит добавляемое по умолчанию значение

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PECancelPrintJob** — эта функция может быть привязана к кнопке, меню и т. п. и обеспечивает пользователя механизмом остановки отчета, находящегося в процессе загрузки данных. Вы можете использовать эту команду в случае отключения кнопки **Отмена** в окне просмотра отчета с помощью команды PEShowPrintControls.

```
procedure PECancelPrintJob (
    printJob: Word
    )stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо отменить

Функция **PECanCloseEngine** — определяет возможность закрытия Crystal Report Engine. Функция вызывается перед PECloseEngine для того, чтобы проверить состояние Print Engine. Если работа Crystal Report Engine будет прекращена в момент обработки отчета, могут возникнуть ошибки в работе приложения или операционной системы.

```
function PECanCloseEngine:
    Bool stdcall;
```

Возвращаемые значения:

- TRUE — если Print Engine может быть закрыта;
- FALSE — если Print Engine занята.

Функция **PECheckFormula** проверяет правильность указанной формулы на наличие ошибок.

```
function PECheckFormula (
    printJob: Word;
    formulaName: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, содержащий указанную формулу для проверки
formulaName	Строка, содержащая имя формулы, которую требуется проверить

Возвращаемые значения:

- TRUE — если формула корректна;
- FALSE — если формула имеет ошибки.

Функция **PECheckGroupSelectionFormula** — проверяет текст формулы, устанавливающей ограничения на сгруппированные данные (Group Selection Formula), на наличие ошибок. Используйте эту функцию тогда, когда формула была изменена.

```
function PECheckGroupSelectionFormula (
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, содержащий формулу выборки по группе для проверки

Возвращаемые значения:

- TRUE — если формула не имеет ошибок;
- FALSE — если формула содержит ошибки.

Функция **PECheckNthTableDifferences** — выполняет поиск различий в таблице базы данных и таблице отчета. Эта функция не работает для отчетов, основанных на словарях.

```
function PECheckNthTableDifferences (
    printJob: Smallint;
    tableN: Smallint;
    tabledifferenceinfo: PETableDifferenceInfo): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, содержащий таблицу базы данных, которую необходимо проверить
tableN	Определяет номер таблицы, для которой вы желаете извлечь информацию. Первая таблица имеет номер 0, а последняя N – 1
tabledifferenceinfo	Запись PETableDifferenceInfo, которая будет содержать найденную информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен;
- код ошибки, см. фирменную документацию, или PE_ERR_NOTIMPLEMENTED, если указанный отчет сформирован на основании словаря.

Функция **PECheckSelectionFormula** — выполняет проверку текста формулы, обеспечивающей фильтрацию данных (Record Selection Formula) в отчете.

Используйте эту функцию тогда, когда функция фильтрации данных была изменена и ее необходимо проверить на наличие ошибок.

```
function PECheckSelectionFormula (
    printJob: Word
     ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, содержащий формулу фильтрации данных, которую необходимо проверить

Возвращаемые значения:

- TRUE — если формула фильтрации данных не содержит ошибок;
- FALSE — если формула фильтрации данных содержит ошибки.

Функция **PECheckSQLExpression** — выполняет проверку выбранного SQL-выражения. Используйте эту функцию, когда SQL-выражение было изменено и вам необходимо проверить получившуюся формулу.

```
function PECheckSQLExpression (
    printJob: smallint;
    const expressionName: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, содержащий SQL-выражение, которое необходимо проверить
expressionName	Строка, содержащая имя SQL-выражения, которое необходимо проверить

Возвращаемые значения:

- TRUE — если SQL-выражение не имеет ошибок;
- FALSE — если SQL-выражение содержит ошибки.

Функция **PECloseEngine** — завершает работу Crystal Report Engine. Все активные процессы завершаются, и все окна просмотра закрываются. Эта функция останавливает Crystal Report Engine, но отправленные на печать отчеты продолжат работу.

```
procedure PECloseEngine stdcall;
```

Функция **PEClosePrintJob** — завершает работу выбранного отчета. Если печать на принтер не завершилась, она продолжится; если окно просмотра отчета открыто, оно останется открытым.

```
function PEClosePrintJob (
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо закрыть

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PECloseSubreport** — закрывает выбранный подотчет.

```
function PECloseSubreport (
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Указатель на подотчет, который необходимо закрыть

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PECloseWindow** — закрывает окно предварительного просмотра. Используйте эту функцию для закрытия окна просмотра по событию из приложения либо в случае, когда кнопка <Закрыть> отключена с помощью функции PEShowPrintControls.

```
procedure PECloseWindow (
    printJob: Word
    )stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, окно просмотра которого необходимо закрыть

Функция **PEConvertPFInfotoVInfo** — конвертирует значение, возвращаемое в параметры CurrentValue или DefaultValue, функции PEParameterFieldInfo и записывает результат в PEValueInfo.

```
function PEConvertPFInfoToVInfo (
    value: PChar;
    valueType: Word;
    var valueInfo: PEValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
value	Строка указатель на текущее значение или значение по умолчанию (возвращается из PEGetNthParameterField) для конвертации
valueType	Определитель типа поля параметра, из которого будет получено значение. Используйте одну из констант Parameter Field Value Type Constants:
	PE_PF_NUMBER
	PE_PF_CURRENCY
	PE_PF_BOOLEAN
	PE_PF_DATE
	PE_PF_STRING
	PE_PF_DATETIME
	PE_PF_TIME
valueInfo	Запись PEValueInfo, которая используется для записи значения в конвертированном виде

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEConvertVInfotoPFInfo** — конвертирует значение, записанное в PEValueInfo, в бинарное представление, используемое функцией PESetNthParameterField.

```
function PEConvertVInfoToPFInfo (
    var valueInfo: PEValueInfo;
    var valueType: Word;
    value: Pchar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
valueInfo	Запись PEValueInfo, которая содержит значение для конвертации
valueType	Определитель типа конвертируемого значения
value	Строка, используемая для записи значения в конвертированном виде, для передачи в PESetNthParameterField

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEDeleteNthGroupSortField** — удаляет указанное поле группировки (***Group Sort Field***). Эта функция используется в том случае, когда необходимо обеспечить возможность удаления группировки в момент работы с отчетом. Когда вы предоставляете пользователю возможность удаления группировки в момент печати, необходимо дать возможность изменять значение, которое будет записано в sortFieldN.

Эта функция может быть использована самостоятельно для удаления существующего поля группировки в случае, когда номер нужного поля известен либо в группе функций:

- PEGetNGroupSortFields — вызывается один раз;
- PEGetNthGroupSortField и PEGetHandleString — вызываются совместно столько раз, сколько потребуется для идентификации требуемого поля;
- PEDeleteNthGroupSortField — вызывается один раз, когда требуемое поле идентифицировано.

Комбинация функций может быть использована для идентификации и последующего удаления существующего поля группировки и/или поля сортировки как реакция на выбор пользователя в момент работы с отчетом.

```
function PEDeleteNthGroupSortField (
    printJob: Word;
    sortFieldN: integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо удалить поле группировки
sortFieldN	Номер поля, которое необходимо удалить. Если N = 0, функция удалит первое поле группировки. Если отчет имеет N полей группировки, функция может быть вызвана в диапазоне sortFieldN между 0 и N-1

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEDeleteNthParameterDefaultValue** — удаляет значение по умолчанию для указанного поля параметра в выбранном отчете.

```
function PEDeleteNthParameterDefaultValue (
    printJob: smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    index: smallint
    ): BOOL stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо удалить значение по умолчанию
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя обрабатываемого отчета
index	Индекс-номер значения по умолчанию, которое необходимо удалить

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEDeleteNthSortField** — удаляет указанное условие сортировки, поле сортировки, из списка ***sort order***. Эта функция используется в том случае, когда необходимо удалить условие сортировки в момент работы с отчетом. Когда вы предоставляете пользователю возможность удаления условия сортировки, необходимо дать возможность изменять значение, которое будет записано в sortFieldN.

Эта функция может быть использована самостоятельно для удаления существующего условия сортировки в случае, если номер нужного поля сортировки известен, либо в группе функций:

- PEGetNSortFields — вызывается один раз;
- PEGetHandleString — вызывается столько раз, сколько потребуется для идентификации требуемого поля сортировки;

- `PEDeleteNthSortField` — вызывается один раз, когда требуемое поле идентифицировано.

Комбинация функций может быть использована для идентификации и последующего удаления существующего поля сортировки как реакция на выбор пользователя в момент работы с отчетом.

```
function PEDeleteNthSortField (
    printJob: Word;
    sortFieldN: integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо удалить поле сортировки
sortFieldN	Номер поля, которое необходимо удалить. Если N = 0, функция удалит первое поле сортировки. Если отчет имеет N полей сортировки, функция может быть вызвана в диапазоне sortFieldN между 0 и N – 1

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEDiscardSavedData** — отключает данные, которые были сохранены вместе с отчетом. Если отчет был сохранен с данными, вы можете, используя эту функцию, заставить Crystal Report Engine обновить данные из источника в момент печати или вывода отчета на просмотр.

```
function PEDiscardSavedData (
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо отключить использование сохраненных ранее данных

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEEnableNthAlert** включает или отключает предупреждения отчета.

```
function PEEnableNthAlert(
   printJob : Smallint;
   alertN : Smallint;
   enabled : Boolean
) : Boolean stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо включить или отключить предупреждения
alertN	Номер предупреждения
enabled	Флаг включения или отключения предупреждения. Если TRUE — показывать предупреждение, если FALSE — не показывать

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEEnableEvent** — включает или отключает события. Все события отключены по умолчанию.

```
function PEEnableEvent (
printJob: Word;
Var enableEventInfo: PEEnableEventInfo
): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо включить или отключить события
enableEventInfo	Запись PEEnableEventInfo

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEEnableProgressDialog** — определяет, будет или не будет показано окно прогресса. Окно прогресса показывает процесс чтения и выборки записей.

```
function PEEnableProgressDialog (
    printJob: Word;
```

```
    enable: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется показать или скрыть окно прогресса
enable	Флаг включения или отключения окна прогресса. Если TRUE — окно показывать, если FALSE — не показывать

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEExportPrintWindow** — экспортирует отчет, открытый в окне просмотра, в файл или отправляет по почте (E-Mail). Эта функция может быть использована в случае, когда открытый отчет удовлетворяет пользователя и его необходимо экспортировать, но при этом кнопка **Экспорт** отключена с помощью функции PEShowPrintControls.

```
function PEExportPrintWindow (
    printJob: Word;
    toMail: Bool;
    waitUntilDone: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо экспортировать
toMail	Флаг отправки по почте (E-Mail)
waitUntilDone	Этот параметр всегда должен быть TRUE

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEExportTo** — экспортирует отчет, используя имя, формат и назначение, определяемое структурой PEExportOptions. Данную функцию можно использовать в любой момент, когда требуется получить информацию об отчете в формате, отличном от Crystal Reports.

```
function PEExportTo (
    printJob: Word;
```

```
    var options: PEExportOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо экспортировать
options	Запись PEExportOptions, определяющая параметры экспорта

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetAllowPromptDialog** — определяет необходимость передачи параметров в выбранный отчет.

```
function PEGetAllowPromptDialog (
    printJob: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется определить необходимость передачи параметров

Возвращаемые значения:

- TRUE — если требуется передача параметров;
- FALSE — если передача параметров не требуется.

Функция **PEGetAreaFormat** — возвращает параметры форматирования выбранного раздела отчета и подставляет его как часть структуры PESectionOptions. Используйте эту функцию для изменения параметров форматирования и их передачи обратно в отчет с помощью функции PESetAreaFormat.

```
function PEGetAreaFormat (
    printJob: Word
    areaCode: Integer;
    options: PESectionOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить параметры форматирования

(окончание)

Параметры функции	Описание параметров
areaCode	Специфические коды разделов, для которых необходимо получить параметры форматирования: PE_ALLSECTIONS PE_SECT_PAGE_HEADER PE_SECT_PAGE_FOOTER PE_SECT_REPORT_HEADER PE_SECT_REPORT_FOOTER PE_SECT_GROUP_HEADER PE_SECT_GROUP_FOOTER PE_SECT_DETAIL
options	Запись PESectionOptions, которая будет получать информацию о параметрах форматирования выбранного раздела отчета

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetAreaFormatFormula** — возвращает текст формулы форматирования секции в виде строкового указателя. Используйте эту функцию, если требуется изменить формулу форматирования секции и передать изменения обратно в отчет с помощью функции PESetAreaFormatFormula. Для извлечения текста формулы можно воспользоваться функцией PEGetHandleString.

```
function PEGetAreaFormatFormula (
    printJob: Word;
    areaCode: Word;
    formulaName: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить формулу форматирования раздела

(окончание)

Параметры функции	Описание параметров
areaCode	Специфические коды разделов, для которых необходимо получить параметры форматирования: PE_ALLSECTIONS PE_SECT_PAGE_HEADER PE_SECT_PAGE_FOOTER PE_SECT_REPORT_HEADER PE_SECT_REPORT_FOOTER PE_SECT_GROUP_HEADER PE_SECT_GROUP_FOOTER PE_SECT_DETAIL
formulaName	Константа, определяющая имя формулы форматирования, для которой вы хотите записать новую строку: PE_FFN_AREASECTION_VISIBILITY PE_FFN_SECTION_VISIBILITY PE_FFN_SHOW_AREA PE_FFN_NEW_PAGE_BEFORE PE_FFN_NEW_PAGE_AFTER PE_FFN_KEEP_TOGETHER PE_FFN_SUPPRESS_BLANK_SECTION PE_FFN_RESET_PAGE_N_AFTER PE_FFN_PRINT_AT_BOTTOM_OF_PAGE PE_FFN_UNDERLAY_SECTION PE_FFN_SECTION_BACK_COLOUR PE_FFN_SECTION_BACK_COLOR
textHandle	Указатель на строку, содержащую текст формулы
textLength	Длина строки формулы в байтах, включая заключительный байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetEnableEventInfo** — возвращает информацию о возможном событии. Используйте функцию для того, чтобы определить, какое событие произойдет.

```
function PEGetEnableEventInfo(
    printJob: Word;
    Var enableEventInfo: PEEnableEventInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо получить информацию о возможных событиях
enableEventInfo	Запись PEEnableEventInfo, содержащая информацию о событиях

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetErrorCode** — возвращает номер ошибки, которая может возникнуть при работе функций Crystal Report Engine. Когда функция выполняется с ошибкой, возвращается код, на основании которого можно реализовать необходимую обработку. Функция PEGetErrorCode должна быть вызвана немедленно после возникновения ошибки.

```
function PEGetErrorCode (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, при работе с которым необходимо получать номера ошибок. Если функция вызывается перед тем, как отчет будет выбран и загружен, используйте 0 для данного параметра

Возвращаемые значения:

- номер ошибки, если функция, вызванная ранее, выполнилась с ошибкой;
- 0, если все работает без ошибок.

Функция **PEGetErrorText** — возвращает указатель на строку, содержащую информацию об ошибке, возникшей при выполнении функций Crystal Report

Engine. Эта функция используется совместно с функцией PEGetHandleString. Данную функцию можно использовать для отображения информации об ошибке в окне сообщения, совместно со своим текстом.

```
function PEGetErrorText (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, при работе с которым необходимо получать описание ошибок. Если функция вызывается перед тем, как отчет будет выбран и загружен, используйте 0 для данного параметра
textHandle	Указатель на строку, содержащую текст ошибки
textLength	Длина строки с текстом ошибки в байтах, включая завершающий байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetExportOptions** — возвращает параметры экспорта перед тем, как отчет будет экспортирован. Функция PEGetExportOptions может быть использована для того, чтобы в виде набора диалогов отобразить параметры экспорта, заданные пользователем. Эти параметры используются Crystal Report Engine для заполнения структуры PEExportOptions. В дальнейшем функция PEExportTo может быть использована для того, чтобы установить параметры назначения отчета на основе информации в PEExportOptions.

```
function PEGetExportOptions (
    printJob: Word;
    var options: PEExportOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить параметры экспорта
options	Запись PEExportOptions, содержащая информацию для экспорта

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetFieldMappingType** — возвращает тип планировки полей для указанного отчета.

```
function PEGetFieldMappingType (
    printJob: Smallint;
    var mappingType: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить код планировки полей
mappingType	Константа соответствующего типа PE_FM_XXX:
	PE_FM_AUTO_FLD_MAP
	PE_FM_CRPE_PROMPT_FLD_MAP
	PE_FM_EVENT_DEFINED_FLD_MAP

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetFormula** — возвращает текст выбранной формулы как указатель на строку. Эта функция используется совместно с PEGetHandleString. Используйте функцию PESetFormula для того, чтобы передать текст формулы обратно. Этот набор функций может быть использован для идентификации и дальнейшего изменения существующей в отчете формулы.

```
function PEGetFormula (
    printJob: Word;
    formulaName: PChar;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо извлечь текст формулы

(окончание)

Параметры функции	Описание параметров
formulaName	Null-завершенная строка, содержащая имя формулы, для которой необходимо извлечь текст
textHandle	Указатель на строку, содержащую текст формулы
textLength	Длина строки в байтах, включая завершающий байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если указанная формула не существует в отчете.

Функция **PEGetFormulaSyntax** — возвращает информацию о синтаксисе Crystal/Basic, использовавшемся при написании формулы, к которой обращались последний раз через функции API.

```
procedure PEGetFormulaSyntax (
    printJob: Word;
    var formulaSyntax: PEFormulaSyntax
    ):Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо определить тип синтаксиса
formulaSyntax	Запись PEFormulaSyntax, которая будет содержать извлекаемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGraphAxisInfo** — возвращает параметры осей для графика, если они доступны.

```
function PEGetGraphAxisInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    var graphAxisInfo : PEGraphAxisInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить параметры осей для графика
sectionN	Указатель на секцию отчета, содержащую график, для которого необходимо получить информацию об осях координат
graphN	Номер графика, для которого требуется получить информацию об осях координат. Номер первого графика 0. Графики нумеруются в порядке их помещения в отчет
graphAxisInfo	Запись PEGraphAxisInfo, которая будет содержать возвращаемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGraphFontInfo** — возвращает информацию о шрифте, который используется в указанном графике.

```
function PEGetGraphFontInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    titleFontType : Word;
    var fontColourInfo : PEFontColorInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о параметрах шрифта, установленного для графика
sectionN	Номер секции, в которой находится график. Этот параметр должен совпадать с диапазоном, возвращаемым из функции PEGetNSections
graphN	Номер графика, для которого требуется получить информацию об осях координат. Номер первого графика 0. Графики нумеруются в порядке их размещения в отчет
titleFontType	Используются константы типа PE_GTF_XXX: PE_GTF_TITLEFONT PE_GTF_SUBTITLEFONT PE_GTF_FOOTNOTEFONT

(окончание)

Параметры функции	Описание параметров
titleFontType	Используются константы типа PE_GTF_XXX: PE_GTF_GROUPSTITLEFONT PE_GTF_DATATITLEFONT PE_GTF_LEGENDFONT PE_GTF_GROUPLABELSFONT PE_GTF_DATALABELSFONT
fontColourInfo	Запись PEFontColorInfo, которая будет содержать возвращаемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGraphOptionInfo** — возвращает параметры отображения графика.

```
function PEGetGraphOptionInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    var graphOptionInfo : PEGraphOptionInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о параметрах отображения, установленных для графика
sectionN	Номер секции, в которой находится график. Этот параметр должен совпадать с диапазоном, возвращаемым из функции PEGetNSections
graphN	Номер графика. Номер первого графика 0. Графики нумеруются в порядке их размещения в отчет
graphOptionInfo	Запись PEGraphOptionInfo, которая будет содержать возвращаемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGraphTextDefaultOption** — определяет, требуется или нет отображать заголовок графика, заданный по умолчанию.

```
procedure PEGetGraphTextDefaultOption (
    printJob: Word;
    sectionN: Smallint;
    graphN: Smallint;
    titleType: Word;
    var useDefault: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о заголовке графика по умолчанию
sectionN	Номер секции, в которой находится график. Этот параметр должен совпадать с диапазоном, возвращаемым из функции PEGetNSections
graphN	Номер графика. Нумерация начинается с 0 и заканчивается N-1. Графики нумеруются в порядке их размещения в отчет
titleType	Константа, определяющая тип заголовка. Используется один из вариантов PE_GTT_XXX: PE_GTT_TITLE PE_GTT_SUBTITLE PE_GTT_FOOTNOTE PE_GTT_SERIESTITLE PE_GTT_GROUPSTITLE PE_GTT_XAXISTITLE PE_GTT_YAXISTITLE PE_GTT_ZAXISTITLE
useDefault	Логический флаг, указывающий на необходимость отображения заголовка графика по умолчанию. TRUE — необходимо показать, FALSE — показывать не надо

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGraphTextInfo** — возвращает текст заголовка для выбранного графика. Эта функция используется совместно с функцией PEGetHandleString.

```
function PEGetGraphTextInfo (
    printJob: Word;
    sectionN: Smallint;
    graphN: Smallint;
    titleType: Word;
    var title: Hwnd;
    var titleLength: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить текст заголовка графика
sectionN	Номер секции, в которой находится график. Этот параметр должен совпадать с диапазоном, возвращаемым из функции PEGetNSections
graphN	Номер графика. Нумерация начинается с 0 и заканчивается N-1. Графики нумеруются в порядке их размещения в отчет
titleType	Тип заголовка. Используются константы типа PE_GTT_XXX:
	PE_GTT_TITLE
	PE_GTT_SUBTITLE
	PE_GTT_FOOTNOTE
	PE_GTT_SERIESTITLE
	PE_GTT_GROUPSTITLE
	PE_GTT_XAXISTITLE
	PE_GTT_YAXISTITLE
	PE_GTT_ZAXISTITLE
title	Указатель на заголовок графика
titleLength	Длина заголовка

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGraphTypeInfo** — возвращает информацию о типе указанного графика.

```
function PEGetGraphTypeInfo (
    printJob: Word;
    sectionN: Smallint;
    graphN: Smallint;
    var graphTypeInfo: PEGraphTypeInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о типе графика
sectionN	Номер секции, в которой находится график. Этот параметр должен совпадать с диапазоном, возвращаемым из функции PEGetNSections
graphN	Номер графика. Нумерация начинается с 0 и заканчивается N-1. Графики нумеруются в порядке их размещения в отчет
graphTypeInfo	Запись PEGraphTypeInfo, в которой будет содержаться возвращаемая информация

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGroupCondition** — определяет условие, по которому выполняется группирование в указанном отчете. Используется для определения текущего условия группирования и его изменения с помощью функции PESetGroupCondition, если это необходимо.

```
function PEGetGroupCondition (
    printJob: Word;
    sectionCode: Integer;
    var conditionFieldHandle: HWnd;
    var conditionFieldLength: Word;
    var condition: Word;
    var sortDirection: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию об условиях группировки
sectionCode	Код секции, для которой необходимо получить информацию об условии группировки: PE_ALLSECTIONS PE_SECT_PAGE_HEADER PE_SECT_PAGE_FOOTER PE_SECT_REPORT_HEADER PE_SECT_REPORT_FOOTER PE_SECT_GROUP_HEADER PE_SECT_GROUP_FOOTER PE_SECT_DETAIL
conditionFieldHandle	Указатель на поле, содержащее условие группировки
conditionFieldLength	Указатель длины поля, содержащего условие группировки
condition	Возвращаемый параметр. Декодирует условие и тип поля. Можно использовать маску для условия PE_GC_CONDITIONMASK или для типа PE_GC_TYPEMASK. При необходимости можно объединить маски с помощью оператора И
sortDirection	Условие сортировки, установленное на группу. Используются следующие константы: Константа PE_SF_DESCENDING Описание: сортировка данных в порядке убывания (от Z к A, от 9 к 1) Константа PE_SF_ASCENDING Описание: сортировка данных в порядке возрастания (от A к Z, от 1 к 9) Константа PE_SF_ORIGINAL Описание: сортировка данных в том порядке, как они записаны в базу данных Константа PE_SF_SPECIFIED Описание: сортировка данных в заданном порядке с дополнительными условиями

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGroupOptions** — используется для определения текущих параметров группировки в отчете.

```
function PEGetGroupOptions (
    printJob: Word;
    groupN: Smallint;
    var groupOptions: PEGroupOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о параметрах группировки
groupN	Номер группы. Нумерация начинается с нуля
groupOptions	Запись PEGroupOptions

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetGroupSelectionFormula** — возвращает строку, содержащую текст формулы, по которой осуществляется группировка в отчете. Эта функция используется совместно с PEGetHandleString. Для изменения формулы необходимо использовать PESetGroupSelectionFormula. Серия этих функций может быть использована для идентификации и изменения формулы группировки в отчете в момент работы с ним из приложения.

```
function PEGetGroupSelectionFormula (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить текст формулы
textHandle	Указатель на строку, содержащую текст формулы

(окончание)

Параметры функции	Описание параметров
textLength	Длина строки, содержащей текст формулы в байтах, включая завершающий байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция `PEGetHandleString` — возвращает текст, который связан с указателем типа `textHandle`. Буфер получит текущий текст. Эта функция используется совместно с функцией, возвращающей длину строки. После того как программа выделит буфер соответствующего размера, функция извлечет строку, связанную с указателем на этот буфер.

```
function PEGetHandleString (
    textHandle: HWnd;
    buffer: PChar;
    bufferLength: Integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
textHandle	Указатель на строку, содержащую интересующий текст
buffer	Буфер, в который копируется интересующий текст
bufferLength	Длина буфера в байтах, включающая null-байт завершения строки. Это значение должно быть идентичным длине строки возвращаемой в переменную функции, определяющей эту длину

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция `PEGetJobStatus` — возвращает текущее состояние отчета, открытого с помощью функции PEOpenPrintJob. С помощью этой функции можно обработать ряд программных событий. Например:

- для отслеживания сообщений об ошибках таких, как недостаточный объем памяти, недостаточное место на диске и т. п.;

- для обработки состояния на экране (песочные часы, различные графические серии и т. п.), чтобы уведомить пользователя о том, что процесс находится в стадии выполнения;
- для поиска сомнительных моментов, из-за которых пользователем был остановлен процесс выполнения отчета после выполнения функции PEStartPrintJob.

```
function PEGetJobStatus (
    printJob: Word;
    var jobInfo: PEJobInfo
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо получить статус
jobInfo	Запись PEJobInfo, которая содержит информацию, возвращаемую функцией

Возвращаемые значения:

- возвращает 0, если функция PEOpenEngine или PEOpenPrintJob вызваны некорректно;
- иначе возвращает одну из констант Job Status Constants.

Константы Job Status Constants	Описание константы
PE_JOBNOTSTARTED	Отчет не запущен
PE_JOBINPROGRESS	Отчет в процессе обработки
PE_JOBCOMPLETED	Обработка отчета завершена
PE_JOBFAILED	Ошибка
PE_JOBCANCELLED	Обработка отчета отменена пользователем
PE_JOBHALTED	Обработка отчета прекратилась из-за большого количества записей или слишком большого времени ожидания

Функция **PEGetMargins** — возвращает параметры границ для страницы отчета. Данная функция используется совместно с функцией PESetMargins для изменения ранее заданных параметров границ отчета.

```
function PEGetMargins (
    printJob: Word;
    var left: Word;
```

```
    var right: Word;
    var top: Word;
    var bottom: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо получить параметры границ
left	Значение параметра для левой границы
right	Значение параметра для правой границы
top	Значение параметра для верхней границы
bottom	Значение параметра для нижней границы

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNDetailCopies** — возвращает количество копий секции Detail в отчете, которое будет распечатано. Эта функция используется для получения информации о том, сколько раз каждая секция **Detail** отчета будет распечатана. Чтобы изменить это количество, используется функция PESetNDetailCopies.

```
function PEGetNDetailCopies(
    printJob: Word;
    var nDetailCopies: Integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, к которому посылается запрос на определение количества распечатываемых секций **Detail**
nDetailCopies	Количество распечатываемых копий секции **Detail**

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNFormulas** — возвращает номер формулы в указанном отчете. Чтобы вызвать формулу, обратившись к ней по номеру, воспользуйтесь функцией PEGetNthFormula.

```
function PEGetNFormulas(
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо определить номер содержащейся в нем формулы

Возвращаемые значения:

- номер формулы в отчете;
- 1 — если возникла ошибка.

Функция **PEGetNGroups** возвращает номер группы в отчете.

```
function PEGetNGroups (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить номер указанной группы

Возвращаемые значения:

- номер группы в отчете;
- 1 — если возникла ошибка.

Функция **PEGetNGroupSortFields** — возвращает номер поля `group sort field` в указанном отчете. Эта функция обычно используется совместно с другими. PEGetNGroupSortFields вызывается один раз; PEGetNthGroupSortField и PEGetHandleString вызываются столько раз, сколько это необходимо для идентификации и коррекции `group sort field`, и PESetNthAlertConditionFormula вызывается один раз, когда коррекция завершена. Эта серия функций может быть использована для изменения параметров отчета в момент его вывода на экран или на печать.

```
function PEGetNGroupSortFields (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого вы хотите получить номер поля **sort field**

Возвращаемые значения:

- номер поля **group sort field**;
- 0 — если такого поля нет;
- 1 — в случае возникновения ошибки.

Функция **PEGetNPages** — возвращает количество страниц в отчете. Эта информация может потребоваться в случае, когда желательно вывести указанную страницу на экран с помощью функции PEShowNthPage.

```
function PEGetNPages (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо определить количество страниц

Возвращаемые значения:

- количество страниц в отчете;
- 1 — если возникла ошибка.

Функция **PEGetNParameterCurrentRanges** — возвращает количество значений, определяющих диапазон для полей параметров в отчете.

```
procedure PEGetNParameterCurrentRanges (
    printJob: Smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    ): Word stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого вы хотите получить текущее значение параметра, определенное как диапазон
parameterFieldName	Имя поля-параметра в указанном отчете
reportName	Имя обрабатываемого отчета

Возвращаемые значения:

- количество значений, определяющих диапазон, связанных с указанным полем параметром;
- 1 — если возникает ошибка.

Функция **PEGetNParameterCurrentValues** — возвращает количество значений, которые в текущий момент записаны в указанное поле параметр отчета.

```
function PEGetNParameterCurrentValues (
    printJob: Smallint;
    const parameterFieldName: PChar;
    const reportName: PChar
    ): Word stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которго необходимо определить количество значений в поле параметре
parameterFieldName	Имя поля параметра в указанном отчете
reportName	Имя обрабатываемого отчета

Возвращаемые значения:

- количество значений, записанных в указанное поле параметр;
- 1 — в случае возникновения ошибки.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PEGetNParameterDefaultValues** — возвращает количество значений, по умолчанию связанных с полем параметром в указанном отчете.

```
function PEGetNParameterDefaultValues (
    printJob: Smallint;
    const parameterFieldName: PChar;
    const reportName: PChar
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо определить количество значений по умолчанию, связанных с полем параметром
parameterFieldName	Имя поля параметра в указанном отчете
reportName	Имя обрабатываемого отчета

Возвращаемые значения:

- количество значений по умолчанию;
- 1 — в случае возникновения ошибки.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PEGetNParameterFields** — возвращает количество полей параметров в указанном отчете, включая поля параметры, входящие в подотчеты.

```
function PEGetNParameterFields (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить количество полей параметров

Возвращаемые значения:

- количество полей параметров в отчете;
- 1 — в случае возникновения ошибки.

Функция **PEGetNReportAlerts** — возвращает количество предупреждений (Report Alerts) в указанном отчете.

```
function PEGetNReportAlerts (
    printJob : Smallint
    ) : Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о количестве предупреждений

Возвращаемые значения:

- количество предупреждений в случае удовлетворительного вызова отчета;
- 1 — если отчет вызван с ошибкой.

Функция **PEGetNSections** — возвращает количество секций в указанном отчете. По умолчанию в стандартном отчете имеется пять секций (Report Header, Page Header, Details, Report Footer и Page Footer). Таким образом, функция для простого отчета вернет 5. В случае добавления в отчет группировки или других секций результат увеличивается на соответствующую величину.

```
function PEGetNSections (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить данные о количестве секций

Возвращаемые значения:

- количество секций в отчете;
- 1 — если вызов отчета произошел с ошибкой.

Функция **PEGetNSectionsInArea** — возвращает количество секций в указанной области отчета.

```
function PEGetNSectionsInArea (
    printJob : Word;
    areaCode : Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо определить количество секций в указанной области
areaCode	Номер области, для которой необходимо получить информацию о количестве секций

Возвращаемые значения:

- количество секций в указанной области отчета, если отчет вызван без ошибок;
- 1 — если отчет вызван с ошибкой.

Функция **PEGetNSortFields** — возвращает количество полей, по которым выполняется сортировка в указанном отчете. Эта функция обычно используется совместно с функциями:

- `PEGetNSortFields` — вызывается один раз;
- `PEReportAlertInfo` и `PEGetHandleString` — вызываются столько раз, сколько потребуется;
- `PESetNthSortField` — вызывается один раз для коррекции идентификатора поля сортировки.

Данная комбинация функций может быть использована для уточнения и модификации параметров сортировки в момент печати отчета.

```
function PEGetNSortFields (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо определить количество полей сортировки

Возвращаемые значения:

- количество полей сортировки;
- 0 — если полей сортировки нет;
- 1 — если отчет вызван с ошибкой.

Функция **PEGetNSQLExpressions** — возвращает количество SQL-выражений в указанном отчете.

```
function PEGetNSQLExpressions (
    printJob: Smallint
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо определить количество SQL-выражений

Возвращаемые значения:

- количество SQL-выражений в отчете;
- 1 — при вызове отчета с ошибкой.

Функция **PEGetNSubreportsInSection** — возвращает количество подотчетов в указанной секции отчета.

```
function PEGetNSubreportsInSection (
    printJob: Word;
    sectionCode: Smallint
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
	Номер отчета, из которого необходимо извлечь информацию о количестве подотчетов в указанной секции
sectionCode	Код секции, в которой необходимо вычислить количество подотчетов. Для указания кода используются константы: PE_ALLSECTIONS PE_SECT_PAGE_HEADER PE_SECT_PAGE_FOOTER PE_SECT_REPORT_HEADER PE_SECT_REPORT_FOOTER PE_SECT_GROUP_HEADER PE_SECT_GROUP_FOOTER PE_SECT_DETAIL

Возвращаемые значения:

- количество подотчетов в указанной секции;
- 1 — при вызове отчета с ошибкой.

Замечание

Параметр sectionCode может быть получен с помощью функции PEGetSectionCode.

12 Зак. 1021

Функция **PEGetNTables** — возвращает количество таблиц в открытом отчете. Можно использовать данную функцию для отчетов, созданных как на основании настольных источников данных, так и на основании реляционных.

```
function PEGetNTables (
    printJob: Word
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которго необходимо получить информацию о количестве таблиц

Возвращаемые значения:

- количество таблиц, используемых в отчете;
- 1 — при вызове отчета с ошибкой.

Замечание

Эта функция может работать с любыми источниками данных (Paradox, Xbase, SQL Server, Oracle, Netware и т. п.).

Функция PEGetNTables должна быть вызвана после PEOpenPrintJob и перед PEStartPrintJob.

Функция **PEGetNthAlertInstanceInfo** — возвращает запись PEAlertInstanceInfo, ассоциированную с выбранным предупреждением (Report Alert).

```
function PEGetNthAlertInstanceInfo(
    printJob : Smallint;
    alertN : Smallint;
    instanceN : DWord;
    var alertInstanceInfo : PEAlertInstanceInfo
    ) : Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о предупреждении
alertN	Номер предупреждения, из которого необходимо получить информацию
instanceN	Номер экземпляра предупреждения, из которого необходимо получить информацию
alertInstanceInfo	Запись PEAlertInstanceInfo, содержащая требуемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthFormula** — возвращает информацию об указанной формуле в отчете. Эта функция извлекает имя формулы и ее текст из отчета. Используя данную функцию совместно с PESetFormula, можно выполнять редактирование и полное изменение формулы в отчете.

```
function PEGetNthFormula (
    printJob: Word;
    formulaN: Integer;
    var nameHandle: HWnd;
    var nameLength: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо извлечь информацию по формуле
formulaN	Номер формулы в отчете, из которой необходимо извлечь информацию (нумерация формул начинается с 0)
nameHandle	Указатель на строку, содержащую имя поля
nameLength	Длина строки, содержащей имя формулы, в байтах, включая завершающий байт
textHandle	Строка, содержащая текст формулы
textLength	Длина строки, содержащей текст формулы, в байтах, включая завершающий байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthGroupSortField** — возвращает информацию о выбранном поле, обеспечивающем группировку и сортировку в указанном отчете. Эта функция используется совместно с PEGetHandleString. PEGetNthGroupSortField возвращает имя поля и порядок сортировки (по возрастанию или убыванию). Также эта функция используется совместно с серией других:

- `PEGetNGroupSortFields` вызывается один раз;

- PEGetNthGroupSortField и PEGetHandleString вызываются столько раз, сколько необходимо для корректировки поля, обеспечивающего группировку и сортировку;
- PESetNthAlertConditionFormula вызывается тогда, когда выполняется корректировка параметров сортировки.

```
function PEGetNthGroupSortField (
    printJob: Word;
    sortFieldN: Integer;
    var nameHandle: HWnd;
    var nameLength: Word;
    var direction: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о поле, выполняющем группировку и сортировку данных
sortFieldN	Номер поля, о котором необходимо получить данные. Нумерация полей начинается с 0
nameHandle	Указатель на строку, содержащую имя поля
nameLength	Длина строки, содержащей имя поля, в байтах, включая завершающий байт
direction	Порядок сортировки. Используются следующие константы:
	Константа PE_SF_DESCENDING
	Описание: сортировка данных в порядке убывания (от Z до А, от 9 до 1)
	Константа PE_SF_ASCENDING
	Описание: сортировка данных в порядке возрастания (от А до Z, от 1 до 9)
	Константа PE_SF_ORIGINAL
	Описание: только для полей группировки. Сортировка данных в оригинальном порядке
	Константа PE_SF_SPECIFIED
	Описание: только для полей группировки. Сортировка данных в специальном порядке. Только для чтения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthParameterCurrentRange** — возвращает характеристики диапазона из выбранного поля параметра в указанном отчете.

```
procedure PEGetNthParameterCurrentRange (
    printJob: Smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    index: Smallint;
    var rangeStart: PEValueInfo;
    var rangeEnd: PEValueInfo;
    rangeInfo: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить характеристики диапазона
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
index	Индекс параметра, для которого извлекается диапазон
rangeStart	Запись PEValueInfo, в которую возвращается начальное значение диапазона
rangeEnd	Запись PEValueInfo, в которую возвращается конечное значение диапазона
rangeInfo	Константы, которые определяют правила обработки диапазона: Константа PE_RI_INCLUDEUPPERBOUND Значение 1 Константа PE_RI_INCLUDELOWERBOUND Значение 2 Константа PE_RI_NOUPPERBOUND Значение 4 Константа PE_RI_NOLOWERBOUND Значение 8

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PEGetNthParameterCurrentValue** — возвращает текущее значение параметра в указанном отчете.

```
function PEGetNthParameterCurrentValue (
    printJob: Smallint;
    const parameterFieldName: PChar;
    const reportName: PChar;
    index: Smallint;
    var currentValue: PEValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо определить текущее значение параметра
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
index	Индекс параметра, для которого извлекается значение
currentValue	Запись PEValueInfo, в которую значение параметра вернется. Если значение отсутствует, будет возвращена константа PE_VI_NOVALUE

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PEGetNthParameterDefaultValue** — возвращает значение, заданное по умолчанию, для параметра в указанном отчете.

```
function PEGetNthParameterDefaultValue (
    printJob: Smallint;
    const parameterFieldName: PChar;
    index: Smallint;
    var valueInfo: PEValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо определить значение параметра по умолчанию
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
index	Индекс параметра, для которого извлекается значение
valueInfo	Запись PEValueInfo, содержащая информацию о запрашиваемом значении по умолчанию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PEGetNthParameterField** — возвращает информацию об одном из полей параметров в указанном отчете. Эта функция возвращает имя поля, тип данных и информацию о значении поля. Эта функция чаще всего используется совместно с другими:

- PEGetNParameterFields вызывается один раз;
- PEGetNthParameterField вызывается столько раз, сколько это необходимо для корректировки поля параметра;

- PESetNthParameterField вызывается один раз для коррекции идентификатора поля параметра.

```
function PEGetNthParameterField (
    printJob: Word;
    varN: Smallint;
    var varInfo: PEParameterFieldInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который содержит поле параметр, о котором необходимо получить информацию
parameterN	Номер параметра, о котором необходимо получить информацию
parameterInfo	Запись PEParameterFieldInfo, которая используется для записи полученной информации

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthParameterType** — возвращает тип указанного параметра.

```
function PEGetNthParameterType (
    printJob: Smallint;
    index: Smallint
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о типе параметра
index	Индекс поля параметра

Возвращаемые значения:

- PE_PO_REPORT — отчет;
- PE_PO_STOREDPROC — хранимая процедура;
- PE_PO_QUERY — запрос;
- −1 — индекс неверен.

Функция **PEGetNthParameterValueDescription** — возвращает текст подсказки для параметра.

```
function PEGetNthParameterValueDescription (
    printJob : Smallint;
    parameterFieldName : PChar;
    reportName : PChar;
    index : Smallint;
    var valueDesc : HWnd;
    var valueDescLength : Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо извлечь описание для указанного параметра
parameterFieldName	Строка, содержащая имя поля параметра, для которого необходимо получить описание
reportName	Строка, содержащая имя отчета
index	Индекс параметра, для которого извлекается значение
valueDesc	Указатель на описание, которое было возвращено
valueDescLength	Длина строки, содержащей описание параметра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PEGetNthReportAlert** — возвращает запись (структуру) PEReportAlertInfo из отчета. Запись PEReportAlertInfo содержит информацию об определенном предупреждении (Report Alert).

```
function PEGetNthReportAlert (
    printJob : Smallint;
    alertN : Smallint;
    var reportAlertInfo : PEReportAlertInfo
    ) : Boolean stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо извлечь информацию о предупреждении
alertN	Номер экземпляра предупреждения (Report Alert), для которого необходимо получить информацию
reportAlertInfo	Запись PEAlertInstanceInfo

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthSortField** — возвращает информацию об одном из полей, по которому выполняется сортировка в указанном отчете. Эта функция возвращает имя поля, по которому выполняется сортировка, и направление сортировки (по возрастанию или по убыванию). Имя поля возвращается как указатель на строку. Данная функция обычно используется совместно с другими:

- PEGetNSortFields вызывается один раз;
- PEGetNthSortField и PEGetHandleString вызываются совместно столько раз, сколько это необходимо;
- PESetNthSortField вызывается один раз для выполнения коррекции поля, обеспечивающего сортировку.

```
function PEGetNthSortField (
    printJob: Word;
    sortFieldN: Integer;
    var nameHandle: HWnd;
    var nameLength: Word;
    var direction: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо получить информацию о поле, по которому выполняется сортировка
sortFieldN	Номер поля, для которого необходимо получить информацию. Нумерация полей начинается с 0
nameHandle	Указатель на строку, содержащую имя поля
nameLength	Длина строки, содержащей имя поля, в байтах, включая завершающий байт

(окончание)

Параметры функции	Описание параметров
direction	Константа, определяющая направление сортировки: Константа PE_SF_DESCENDING Описание — сортировка данных в порядке убывания (от Z до A, от 9 до 1) Константа PE_SF_ASCENDING Описание — сортировка данных в порядке возрастания (от A до Z, от 1 до 9) Константа PE_SF_ORIGINAL Описание — только для полей группировки. Сортировка данных в оригинальном порядке Константа PE_SF_SPECIFIED Описание — только для полей группировки. Сортировка данных в специальном порядке. Только для чтения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthSQLExpression** — возвращает SQL-выражение, включенное в отчет. Эта функция используется совместно с PEGetHandleString.

```
function PEGetNthSQLExpression (
    printJob: Smallint;
    expressionN: Smallint;
    var nameHandle: HWnd;
    var nameLength: Smallint;
    var textHandle: HWnd;
    var textLength: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию об SQL-выражении
expressionN	Номер, указывающий на выражение, которое необходимо извлечь
nameHandle	Указатель на строку, содержащую имя SQL-выражения

(окончание)

Параметры функции	Описание параметров
nameLength	Длина строки, содержащей имя SQL-выражения, в байтах, включая завершающий байт
textHandle	Указатель на строку, содержащую SQL-выражение
textLength	Длина строки, содержащей SQL-выражение

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthSubreportInSection** — возвращает указатель, который необходим для получения имени подотчета.

```
function PEGetNthSubreportInSection (
    printJob: Word;
    sectionCode: Smallint;
    subreportN: Smallint
    ): DWord stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который является основным
sectionCode	Константа, указывающая на секцию отчета, в которой находится подотчет:
	PE_ALLSECTIONS
	PE_SECT_PAGE_HEADER
	PE_SECT_PAGE_FOOTER
	PE_SECT_REPORT_HEADER
	PE_SECT_REPORT_FOOTER
	PE_SECT_GROUP_HEADER
	PE_SECT_GROUP_FOOTER
	PE_SECT_DETAIL
subreportN	Номер подотчета в указанной секции. Нумерация подотчетов начинается с 0. Если подотчетов нет в указанной секции, функция вернет 0

Возвращаемые значения: указатель, который используется для получения имени интересуемого подотчета.

Замечание

Используйте PEGetSubreportInfo, чтобы получить информацию о подотчете, указатель для которого вернет функция PEGetNthSubreportInSection.

Функция **PEGetNthTableLocation** — определяет местонахождение таблицы используемой в отчете. Эта функция обычно используется совместно с PESetNthTableLocation для внесения изменений в настройки отчета, связанные с местонахождением таблиц.

```
function PEGetNthTableLocation(
    printJob: Word;
    tableN: Integer;
    var location: PETableLocation
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо извлечь информацию о местоположении таблиц, используемых для формирования набора данных для отчета
tableN	Номер таблицы, используемой при формировании отчета. Нумерация таблиц начинается с 0
location	Запись PETableLocation, содержащая данные о таблице. Формат строки зависит от типа источника, используемого для формирования отчета

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthTableLogOnInfo** — возвращает информацию об имени сервера, имени базы данных, имени пользователя и другую информацию, необходимую для подключения отчета к источнику данных.

```
function PEGetNthTableLogOnInfo (
    printJob: Word;
    tableN: Integer;
    var logOnInfo: PELogOnInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о том, как подключаться к таблице в источнике данных

(окончание)

Параметры функции	Описание параметров
tableN	Номер таблицы, используемой для построения отчета. Нумерация таблиц начинается с 0
logOnInfo	Запись PELogOnInfo, содержащая всю необходимую для подключения к базе данных информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Эта функция должна вызываться после PEOpenPrintJob, так как для нее необходим номер отчета, и перед PEStartPrintJob, для которой может потребоваться указать пароль для подключения к базе данных. Строка пароля, хранимая в записи (структуре) PELogOnInfo, будет всегда пустой при использовании данной функции.

Функция **PEGetNthTablePrivateInfo** — возвращает информацию об использовании таких объектов, как ADO, DAO, RDO или CDO с соответствующими драйверами Active Data Drivers (crdb_ado.dll, crdb_dao.dll, crdb_odbc.dll, crdb_cdo.dll).

```
function PEGetNthTablePrivateInfo (
    printJob: Word;
    tableN: Smallint;
    var privateInfo: PETablePrivateInfo
    ) :Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который использует таблицы, требующие информацию о сессии связи с источником
tableN	Номер таблицы, для которой необходимо получить информацию. Нумерация таблиц начинается с 0
privateInfo	Запись PETablePrivateInfo, содержащая необходимую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetNthTableSessionInfo** — возвращает информацию о сессии для таблиц Microsoft Access, используемых в отчете. Данная функция возвращает следующую информацию: User ID, Password и Session Handle.

```
function PEGetNthTableSessionInfo (
    printJob: Word;
    tableN: Integer;
    var sessionInfo: PESessionInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, требующий получения информации о сессии
tableN	Номер таблицы, используемой в отчете. Нумерация таблиц начинается с 0
sessionInfo	Запись PESessionInfo, которая содержит требуемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Эта функция может быть использована только для баз данных MS Access, которые требуют открытия сессии, прежде чем будет обеспечен доступ к базе. Строка, содержащая пароль, в записи (структуре) PESessionInfo будет всегда пустой.

Функция **PEGetNthTableType** — определяет тип каждой таблицы. Эта функция одна из серии функций, которые обеспечивают возможность получения и изменения параметров работы отчета с базами данных.

```
function PEGetNthTableType (
    printJob: Word;
    tableN: Integer;
    var tableType: PETableType
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо определить тип таблицы

(окончание)

Параметры функции	Описание параметров
tableN	Номер таблицы, входящей в отчет. Нумерация начинается с 0. То есть в случае, когда функция PEGetNTables вернет 2, вызывайте функцию PEGetNthTableType дважды с номерами 0 и 1
tableType	Запись PETableType, возвращающая требуемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Приложение может протестировать DBType в PETableType или библиотеку базы данных DLLName, используемые для создания отчета:

- имена библиотек имеют следующие правила наименования: crdb_*.dll для стандартных (нереляционных) баз данных и SQL/ODBC баз данных;
- в случае использования ODBC-источников (crdb_odbc.dll), и строка DescriptiveName в PETableType содержит имя используемого ODBC DSN.

PEGetNthTableType должна вызываться после PEOpenPrintJob и перед PEStartPrintJob.

Функция **PEGetParameterMinMaxValue** — возвращает возможные максимальное и минимальное значения для указанного параметра в отчете.

```
function PEGetParameterMinMaxValue (
    printJob: Word;
    const parameterFieldName: PChar;
    const reportName: PChar;
    var valueMin: PEValueInfo;
    var valueMax: PEValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо вычислить минимальное и максимальное значение параметра
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета

(окончание)

Параметры функции	Описание параметров
valueMin	Запись PEValueInfo, в которую возвращается информация о минимальном значении параметра
valueMax	Запись PEValueInfo, в которую возвращается информация о максимальном значении параметра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемое значение параметра reportName:

- для основного отчета пустая строка ("");
- для подотчета пути и имени подотчета как строка, завершающаяся NULL.

Ожидаемые значения для параметров valueMin и valueMax:

- установите valueMin в NULL для получения только максимального значения и установите значение, отличное от NULL, если valueMax равно NULL;
- установите valueMax в NULL для получения только минимального значения и установите значение, отличное от -NULL, если valueMin равно NULL.

Функция **PEGetParameterPickListOption** — возвращает список настроек параметра в отчете. Эта функция записывает значения в запись (структуру) PEParameterPickListOption.

```
function PEGetParameterPickListOption (
    printJob: Word;
    parameterFieldName : PChar;
    reportName : PChar;
    var pickListOption : PEParameterPickListOption
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо извлечь список настроек для параметра
parameterFieldName	Строка, содержащая имя параметра, для которого необходимо получить список настроек
reportName	Строка, содержащая имя отчета
pickListOption	Запись PEParameterPickListOption, которая будет содержать возвращаемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемое значение параметра reportName:

- для основного отчета пустая строка ("");
- для подотчета пути и имени подотчета как строка, завершающаяся NULL.

Функция **PEGetParameterValueInfo** — возвращает запись PEParameterValueInfo, связанную с указанным полем параметров в отчете. Эта запись содержит информацию следующего рода:

- возможность редактирования значений параметра;
- допустимость использования значения NULL;
- разрешение использовать множество значений;

и т. п.

```
function PEGetParameterValueInfo (
    printJob : Word;
    const parameterFieldName : PChar;
    const reportNam : PChar;
    var valueInfo : PEParameterValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить информацию о параметре
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
valueInfo	Запись PEParameterValueInfo, в которую будут возвращены интересующие данные

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемое значение параметра reportName:

- для основного отчета пустая строка ("");
- для подотчета пути и имени подотчета как строка, завершающаяся NULL.

Функция **PEGetPrintDate** — определяет дату печати, которая определена в отчете. Используется эта функция для считывания имеющегося значения и передачи нового с помощью функции PESetPrintDate.

```
function PEGetPrintDate (
    printJob: Word;
    var year: Word;
    var month: Word;
    var day: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо узнать дату печати
year	Номер, соответствующий году (4 цифры)
month	Номер, соответствующий месяцу (2 цифры)
day	Номер, соответствующий числу (2 цифры)

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Изменение значения даты печати может потребоваться в случае, когда вы печатаете отчет сегодня, а дата печати должна стоять другая.

Функция **PEGetPrintOptions** — возвращает параметры печати, которые определены для отчета. Эта функция используется для выяснения текущих параметров печати и их последующего изменения с помощью функции PESetPrintOptions.

```
function PEGetPrintOptions (
    printJob: Word;
    var options: PEPrintOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, к которому направляют запрос для выяснения текущих параметров печати
options	Запись PEPrintOptions, в которую возвращаются запрошенные данные

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetReportOptions** — возвращает информацию об основных настройках указанного отчета.

```
function PEGetReportOptions (
    printJob: Word;
    var reportOptions: PEReportOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется узнать информацию об основных настройках
reportOptions	Запись PEReportAlertInfo, содержащая требуемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetReportSummaryInfo** — возвращает суммарную информацию об отчете:

- заголовок отчета;
- автор;
- комментарии.

```
function PEGetReportSummaryInfo (
    printJob: Word;
    var summaryInfo: PEReportSummaryInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, о котором необходимо узнать информацию
summaryInfo	Запись PEReportSummaryInfo, получающая запрашиваемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetReportTitle** — возвращает указатель на строку, содержащую заголовок отчета. Если выбран подотчет, то возвращается указатель на имя подотчета. Функция используется совместно с PEGetHandleString. Для изменения заголовка отчета используется функция PESetReportTitle.

```
function PEGetReportTitle (
    printJob: Word;
    var titleHandle: HWnd;
    var titleLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого необходимо получить данные о заголовке
titleHandle	Указатель на строку, содержащую заголовок отчета
titleLength	Длина строки с заголовком отчета в байтах, включая завершающий байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetReportVersion** — возвращает запись PEVersionInfo, связанную с отчетом. Запись PEVersionInfo содержит информацию о версии отчета.

```
function PEGetReportVersion (
    printJob : Smallint;
    var pVersionInfo : PEVersionInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется информация о версии
pVersionInfo	Запись PEVersionInfo

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetSectionCode** — возвращает код секции. Код секции определяет ее тип (Page Header, Details и т. п.). Если отчет имеет несколько условий группировки, то секции группировки идентифицируются номером группы. В случае наличия множественных секций в одном разделе их идентифицируют по номеру секции.

```
function PEGetSectionCode (
    printJob: Word;
    sectionN: Smallint
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого извлекают код секции
sectionN	Номер секции в отчете, для которой необходимо узнать код. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections

Возвращаемые значения:

- PE_ALLSECTIONS
- PE_SECT_PAGE_HEADER
- PE_SECT_PAGE_FOOTER
- PE_SECT_REPORT_HEADER
- PE_SECT_REPORT_FOOTER
- PE_SECT_GROUP_HEADER
- PE_SECT_GROUP_FOOTER
- PE_SECT_DETAIL
- 0 — если при вызове функции произошла ошибка

Функция **PEGetSectionFormat** — возвращает параметры форматирования секции для последующего их использования в PESectionOptions. Эта функция используется в случае необходимости изменения параметров форматирования секции с помощью функции PESetSectionFormat.

```
function PEGetSectionFormat (
    printJob: Word;
    sectionCode: Integer;
    var options: PESectionOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо узнать информацию о параметрах форматирования указанной секции
sectionCode	Код секции, для которой требуется получить информацию. Код секции можно получить с помощью функции PEGetSectionCode
options	Запись PESectionOptions, в которую будет записана требуемая информация

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetSectionFormatFormula** — возвращает текущую формулу, обеспечивающую форматирование указанной секции в отчете. Данную функцию необходимо использовать совместно с PEGetHandleString.

```
function PEGetSectionFormatFormula (
    printJob: Word;
    sectionCode: Word;
    formulaName: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо узнать формулу форматирования выбранной секции
sectionCode	Код секции, который может быть получен с помощью функции PEGetSectionCode

(окончание)

Параметры функции	Описание параметров
formulaName	Имя формулы, обеспечивающей форматирование. Используются константы: Константа PE_FFN_AREASECTION_VISIBILITY Описание — формула форматирования области отчета и отдельной секции Константа PE_FFN_SECTION_VISIBILITY Описание — формула форматирования секции Константа PE_FFN_SHOW_AREA Описание — формула форматирования области отчета Константа PE_FFN_NEW_PAGE_BEFORE Описание — формула форматирования области отчета и отдельной секции Константа PE_FFN_NEW_PAGE_AFTER Описание — формула форматирования области отчета и отдельной секции Константа PE_FFN_KEEP_TOGETHER Описание — формула форматирования области отчета и отдельной секции Константа PE_FFN_SUPPRESS_BLANK_SECTION Описание — формула форматирования секции Константа PE_FFN_RESET_PAGE_N_AFTER Описание — формула форматирования области отчета и отдельной секции Константа PE_FFN_PRINT_AT_BOTTOM_OF_PAGE Описание — формула форматирования области отчета и отдельной секции Константа PE_FFN_UNDERLAY_SECTION Описание — формула форматирования секции Константа PE_FFN_SECTION_BACK_COLOUR Описание — формула форматирования секции Константа PE_FFN_SECTION_BACK_COLOR Описание — формула форматирования секции
textHandle	Указатель на строку, содержащую текст формулы форматирования
textLength	Длина строки, содержащей текст формулы форматирования. Используйте данное значение для выделения памяти под текст

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Не все имена формул применимы в различных секциях отчета. Используйте значение textLength для выделения памяти под текст. Используйте функцию PEGetHandleString для того, чтобы заполнить выделенный буфер реальным значением.

Функция **PEGetSectionHeight** — возвращает информацию о высоте указанной секции. Вызывается для совместной работы с функцией PEGetMinimumSectionHeight и должна быть использована при создании нового отчета.

```
function PEGetSectionHeight (
    printJob: Word;
    sectionCode: Smallint;
    Height: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого запрашивается информация о высоте секции
sectionCode	Код секции, который может быть получен с помощью функции PEGetSectionCode
height	Высота секции

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetSelectedPrinter** — возвращает информацию о принтере, который в данный момент используется отчетом.

```
function PEGetSelectedPrinter (
    printJob: Word;
    var driverHandle: HWnd;
    var driverLength: Word;
    var printerHandle: HWnd;
```

```
    var printerLength: Word;
    var portHandle: HWnd;
    var portLength: Word;
    var mode: PDeviceModeA
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо получить информацию о принтере
driverHandle	Указатель на драйвер принтера
driverLength	Длина имени драйвера принтера
printerHandle	Указатель на строку, содержащую имя принтера
printerLength	Длина строки, содержащей имя принтера
portHandle	Указатель на порт, к которому подключен принтер
portLength	Длина имени порта
mode	Запись DEVMODE, содержащая информацию о принтере (Windows API)

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetSelectionFormula** — возвращает указатель на строку, содержащую текст формулы, ограничивающей выборку данных в отчет. Используйте PESetSelectionFormula для того, чтобы изменить формулу.

```
function PEGetSelectionFormula (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется проверить и, если потребуется, изменить формулу
textHandle	Указатель на строку, содержащую текст формулы
textLength	Длина строки, содержащей текст формулы, в байтах, включая завершающий байт

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetSQLExpression** — возвращает указанное SQL-выражение в выбранном отчете.

```
function PEGetSQLExpression (
    printJob: Smallint;
    const expressionName: PChar;
    var textHandle: HWnd;
    var textLength: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого требуется извлечь SQL-выражение
expressionName	Строка, содержащая имя выражения
textHandle	Указатель на строку, содержащую текст выражения
textLength	Длина строки, содержащей текст выражения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetSQLQuery** — возвращает SQL-запрос, с помощью которого в отчет извлекаются данные из источника. Используйте функцию PESetSQLQuery для того, чтобы изменить запрос.

```
function PEGetSQLQuery (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, из которого требуется получить информацию о SQL-запросе
textHandle	Указатель на строку, содержащую SQL-запрос
textLength	Длина строки, содержащей SQL-запрос, в байтах, с учетом завершающего байта

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Перед тем как вызывать функцию PEGetSQLQuery отчет должен быть подключен к базе данных.

Функция **PEGetSubreportInfo** — возвращает информацию об указанном подотчете.

```
function PEGetSubreportInfo (
    printJob: Word;
    subreportHandle: DWord;
    var subreportInfo: PESubreportInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который содержит требуемый подотчет
subreportHandle	Указатель на подотчет для которого требуется получить информацию
subreportInfo	Запись PESubreportInfo, которая используется для хранения возвращаемой информации

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetTrackCursorInfo** — возвращает информацию о типе курсора. Различные типы курсоров могут быть использованы для работы в разных областях отчета в режиме просмотра.

```
function PEGetTrackCursorInfo (
    printJob: Smallint;
    var cursorInfo: PETrackCursorInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета , из которого необходимо получить информацию о типе курсора
cursorInfo	Запись PETrackCursorInfo, которая будет содержать информацию о типе курсора

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEGetVersion** — возвращает номер версии DLL или Crystal Report Engine. Эта функция может быть использована тогда, когда требуется реализовать обработку отчетов, которые были сделаны в более ранних версиях Crystal Reports, и требуется уточнение.

```
function PEGetVersion (
    versionRequested: Integer
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
versionRequested	Определяет тип информации — версию DLL или Crystal Report Engine требуется вернуть. Используются следующие константы: Константа PE_GV_DLL Описание — возвращаемые значения: версия DLL (CRPE/CRPE32) Константа PE_GV_ENGINE Описание — возвращаемые значения: версия Crystal Report Engine

Возвращаемые значения: номер версии DLL или Crystal Report Engine.

Функция **PEGetWindowHandle** — возвращает указатель на окно просмотра отчета. Функция PEGetWindowHandle может быть использована для определения состояния отчета — открыт он или нет.

```
function PEGetWindowHandle (
    printJob: Word
    ): HWnd stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется получить информацию об указателе на окно просмотра. Если в момент вызова функции создано несколько окон просмотра, то возвращается информация об окне, созданном самым последним

Возвращаемые значения:

- указатель на окно просмотра — если отчет вызван без ошибок;
- 0 — если при вызове отчета возникла ошибка или окно просмотра уже закрыто.

Замечание

Эта функция может быть использована только после PEStartPrintJob и в том случае, когда вы имеете созданное окно просмотра.

Функция **PEGetWindowOptions** — возвращает информацию о конфигурации окна просмотра отчета.

```
function PEGetWindowOptions (
    printJob: Word;
    var options: PEWindowOptions
    ):Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который открыт в окне просмотра. Если в момент вызова функции открыто несколько отчетов на просмотр, то возвращается информация о последнем созданном окне
PEWindowOptions	Запись PEWindowOptions, в которой будет содержаться информация об окне просмотра отчета

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Когда значение `hasGroupTree` соответствует `True`, это не означает, что в окне просмотра откроется раздел дерева групп. Параметр `hasGroupTree` и свойства отчета `Create Group Tree` в `Crystal Reports` должны совпадать, для того чтобы сделать видимым дерево групп в окне просмотра.

Функция **PEHasSavedData** — возвращает информацию о наличии сохраненных данных в отчете. Используя эту информацию, вы можете определить, требуется обновление данных или нет перед тем, как отчет будет распечатан.

```
function PEHasSavedData(
    printJob: Word;
    var hasSavedData: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, в котором требуется определить наличие сохраненных данных
`hasSavedData`	Индикатор наличия или отсутствия сохраненных данных. TRUE — если данные есть, FALSE — если данных нет

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Отчет может иметь, а может и не иметь сохраненные данные перед печатью или просмотром, на момент печати или просмотра немедленно создается сохраненный набор данных.

Используйте функцию `PEDiscardSavedData` для того, чтобы освободить отчет от связанных с ним сохраненных данных. При последующих вызовах отчета данные будут всегда браться из базы, а не из сохраненных массивов.

Функция **`PEIsPrintJobFinished`** выполняет мониторинг отчета и отслеживает момент завершения работы над отчетом.

```
function PEIsPrintJobFinished (
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, который необходимо проверить

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Функция PEIsPrintJobFinished вернет TRUE сразу же после того, как отчет будет показан в окне просмотра, даже если окно просмотра все еще открыто.

Функция **PELogOffServer** — позволяет выполнить отключение от указанного сервера.

```
function PELogOffServer(
    dllName: PChar;
    var logOnInfo: PELogOnInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
dllName	Имя библиотеки, используемой для подключения к источнику данных
logOnInfo	Запись PELogOnInfo, содержащая информацию о сервере, базе данных, имени и пароле пользователя

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Функции PELogOnServer и PELogOffServer могут быть вызваны в любое время для подключения и отключения от сервера. Эти функции не потребуются, если будет обработана функция PESetNthTableLogOnInfo.

Данная функция требует имя библиотеки, используемой для подключения к базе данных, которое может быть получено из функции PEGetNthTableType.

Функция **PELogOnServer** — используется для подключения отчета к источнику данных.

```
function PELogOnServer(
    dllName: PChar;
    var logOnInfo: PELogOnInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
dllName	Имя библиотеки, используемой для подключения к источнику данных
logOnInfo	Запись PELogOnInfo, содержащая информацию о сервере, базе данных, имени и пароле пользователя

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Функции PELogOnServer и PELogOffServer могут быть вызваны в любое время для подключения и отключения от сервера. Эти функции не потребуются, если будет обработана функция PESetNthTableLogOnInfo.

Данная функция требует имя библиотеки, используемой для подключения к базе данных, которое может быть получено из функции PEGetNthTableType.

Эта функция может быть использована для любых типов источников данных.

Когда отчет обрабатывается с помощью функции PEStartPrintJob, параметр ServerName, используемый в PELogOnServer, не требуется, так как он записан в самом отчете. Функция PELogOnServer может быть использована для переключения отчета на другой сервер в момент вызова отчета из приложения.

Особенности использования функций PELogOnServer и PESetNthTableLogOnInfo:

- PELogOnServer вызывается раньше, чем PESetNthTableLogOnInfo, и может быть вызвана в любое время. Однако для ее работы необходимо знать имя библиотеки, через которую осуществляется подключение к серверу;
- PESetNthTableLogOnInfo более гибкая, чем PELogOnServer. Она позволяет переопределить любые параметры подключения к базе данных. PESetNthTableLogOtInfo должна вызываться после PEOpenPrintJob.

Функция **PELogOnSQLServerWithPrivateInfo** — предоставляет возможность использовать уже существующее подключение к серверу. С помощью данной функции можно снизить количество соединений с базой данных.

```
function PELogOnSQLServerWithPrivateInfo (
    dllName: PChar;
    privateInfo: Pointer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
dllName	Имя библиотеки, используемой для подключения к источнику данных
privateInfo	Указатель на соединение с базой данных. В приложении любое соединение с базой данных формирует указатель на соединение с базой данных (HDBC), содержащий всю необходимую для работы информацию. Этот параметр дает возможность отчетам Crystal Reports соединяться с базой данных по уже существующим каналам

13 Зак 1021

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Если приложение использует ODBC для соединения с базой данных, получить ODBC HDBC можно с помощью следующих функций (так же смотрите документацию по ODBC):

- SQLAllocEnv — выполняется инициализация ODBC на уровне вызова и выделяется память для указателей;
- SQLAllocConnect — возвращается ODBC HDBC.

Функция **PENextPrintWindowMagnification** — используется для изменения размеров отчета в окне просмотра: Full Page, Fit One Side, Fit Both Sides, and Full Page, Fit One Side и т. д. Функция позволяет установить следующий по порядку уровень увеличения.

```
function PENextPrintWindowMagnification (
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который открыт в окне просмотра, и требуется изменить его увеличение

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEOpenEngine** — подготавливает Crystal Report Engine к обработке запросов.

```
function PEOpenEngine
    ): Bool stdcall;
```

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Эта функция вызывается раньше, чем любая другая функция Crystal Report Engine. При возникновении ошибки в момент вызова функции PEOpenEngine может быть использована функция PEGetErrorCode для вывода информации.

Функция PEOpenEngine инициализирует Crystal Report Engine в режиме одинарного потока (single-thread).

Функция **PEOpenPrintJob** — подготавливает отчет для последующей обработки (просмотр, печать и т. д.) и — возвращает номер, который позволяет идентифицировать этот отчет.

```
function PEOpenPrintJob (
    reportFilePath: PChar
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
reportFilePath	Путь и имя к файлу, который вы хотите открыть. Данный параметр записывается в двойных кавычках

Возвращаемые значения:

- номер отчета или, по-другому, указатель на отчет. В других функциях используется как printJob;
- 0 — если файла не существует или возникла ошибка при его обработке.

Замечание

Эта функция вызывается в большинстве случаев раньше, чем все остальные функции Crystal Report Engine.

Одновременно можно обработать только один файл отчета.

Функция PEClosePrintJob должна быть вызвана для закрытия отчета.

Путь к отчету и имя файла должны быть заключены в двойные кавычки:

- PEOpenPrintJob ("C:\CRW\REPORT1.RPT");
- при использовании языков C или C++ значок \ в строке должен быть определен как \\.

Эта функция открывает отчет с указанием принтера, выбранного в отчете, или принтера, принятого по умолчанию.

Функция **PEOpenSubreport** — открывает поименованный подотчет и возвращает идентификационный номер этого подотчета.

```
function PEOpenSubreport (
    parentJob: Word;
    subreportName: PChar
    ): Word stdcall;
```

Параметры функции	Описание параметров
parentJob	Номер отчета, который является основным по отношению к подотчету. Этот указатель возвращается функцией PEOpenPrintJob
subreportName	Указатель на имя подотчета, который необходимо открыть. Эта информация может быть получена с помощью функции PEGetSubreportInfo

Возвращаемые значения:

- номер обрабатываемого подотчета;
- 0 — если подотчет не существует или возникла ошибка при его обработке.

Замечание

Эта функция должна вызываться перед всеми другими функциями Crystal Report Engine, которые обращаются к данному подотчету.

Функция PECloseSubreport используется для закрытия подотчета.

Функция **PEOutputToPrinter** — отправляет подготовленный к работе отчет на выбранный принтер.

```
function PEOutputToPrinter (
    printJob: Word;
    nCopies: Integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо распечатать
nCopies	Количество копий, которое необходимо распечатать. Укажите 0 для того, чтобы использовать установки по умолчанию

Возвращаемые значения:

- TRUE — если вывод на принтер выполнен корректно;
- FALSE — если отчет не может быть выведен на принтер.

Замечание

Если принтер был установлен с помощью функции PESelectPrinter, отчет будет отправлен на этот принтер.

Если функция PESelectPrinter не выполнялась, отчет будет отправлен на тот принтер, который указан непосредственно в самом отчете.

Если функция PESelectPrinter не выполнялась и в отчете не прописан принтер, то отчет будет выведен на тот принтер, который определен в Windows как принтер по умолчанию.

Набор функций, которые помогут вывести отчет на принтер:

- PEOpenPrintJob — открывает отчет;
- PEOutputToWindow — выводит отчет в окно просмотра с заданием параметров окна;
- PEStartPrintJob — выводит отчет в окно просмотра с параметрами по умолчанию;
- PEOutputToPrinter — выводит отчет непосредственно на принтер;
- PESelectPrinter — позволяет переопределить ранее выбранный принтер;
- PEStartPrintJob — после смены принтера открывает окно просмотра отчета с новыми настройками;
- PEClosePrintJob — закрывает отчет.

Функция **PEOutputToWindow** — направляет отчет непосредственно в окно просмотра.

```
function PEOutputToWindow (
    printJob: Word;
    title: PChar;
    left: Longint;
    top: Longint;
    width: Longint;
    height: Longint;
    style: Longint;
    parentWindow: HWnd
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо просмотреть перед печатью
title	Строка, содержащая текст, который должен появиться в заголовке окна просмотра
left	Абсцисса верхнего левого угла окна просмотра в пикселах
top	Ордината верхнего левого угла окна просмотра в пикселах
width	Ширина окна просмотра в пикселах
height	Высота окна просмотра в пикселах

(продолжение)

Параметры функции	Описание параметров
`style`	Стиль окна, которое будет создано. При формировании стиля можно пользоваться комбинацией констант, объединенных оператором OR: Константа `WS_MINIMIZE` Значение 536870912 Описание: создается окно минимального размера Константа `WS_VISIBLE` Значение 268435456 Описание: окно становится видимым сразу же при вызове функции Константа `WS_DISABLED` Значение 134217728 Описание: окно становится недоступным для любых воздействий Константа `WS_CLIPSIBLINGS` Значение 67108864 Описание: закрепляется как дочернее окно относительно другого окна Константа `WS_CLIPCHILDREN` Значение 33554432 Описание: исключается область дочернего окна, попадающая при прорисовке за рамки родительского Константа `WS_MAXIMIZE` Значение 16777216 Описание: создается окно максимального размера Константа `WS_CAPTION` Значение 12582912 Описание: создается окно, содержащее панель заголовка Константа `WS_BORDER` Значение 8388608 Описание: создается окно, окруженное рамкой Константа `WS_DLGFRAME` Значение 4194304 Описание: создается окно, окруженное двойной рамкой и не имеющее панели заголовка Константа `WS_VSCROLL` Значение 2097152 Описание: создается окно, которое содержит панель вертикального скроллинга

(окончание)

Параметры функции	Описание параметров
style	Константа WS_HSCROLL Значение 1048576 Описание: создается окно, которое содержит панель горизонтального скроллинга Константа WS_SYSMENU Значение 524288 Описание: добавляет в окно панель системных кнопок и меню Константа WS_THICKFRAME Значение 262144 Описание: включается широкая рамка, используемая для изменения окна просмотра Константа WS_MINIMIZEBOX Значение 131072 Описание: добавляется кнопка минимизации Константа WS_MAXIMIZEBOX Значение 65536 Описание: добавляется кнопка максимизации Константа CW_USEDFAULT Значение -32768 Описание: устанавливаются параметры, соответствующие значениям по умолчанию для дочерних окон
parentWindow	Указатель на родительское окно, которое выступает носителем окна просмотра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Для самостоятельного окна просмотра верхний левый угол определяется относительно всего экрана. Для дочернего окна в приложении MDI верхний левый угол определяется относительно рабочей области главного окна приложения. Для окна просмотра, которое использует в качестве подложки другое окно, верхний левый угол определяется относительно этого окна.

Если параметр parentWindow равен NULL, окно просмотра является самостоятельным окном верхнего уровня, которое не может быть дочерним по отношению к другим окнам. Для такого варианта окна просмотра можно

использовать константу CW_USDEFAULT, чтобы расположить его в рамки, заданные по умолчанию.

Если в качестве родительского окна определено главное окно MDI-приложения, то окно просмотра будет выведено как дочернее в рабочую область родительского окна.

Если параметр parentWindow указывает на любое другое окно приложения, то окно просмотра становится дочерним для этого окна и размещается внутри него.

Если окно просмотра является самостоятельным или дочерним по отношению к главному окну MDI-приложения и параметр *style* равен 0, то этот стиль заменяется по умолчанию на следующую комбинацию стилевых констант:

(WS_VISIBLE | WS_THICKFRAME | WS_SYSMENU | WS_MAXIMIZEBOX | WS_MINIMIZEBOX)

Эти параметры являются параметрами по умолчанию для любых окон в Windows.

Окно просмотра создается после вызова функции PEStartPrintJob.

Функция **PEPrintControlsShowing** — определяет наличие кнопок, позволяющих выполнять печать и настраивать принтер в окне просмотра отчета. Эта функция используется совместно с **PEShowPrintControls** для изменения видимой комбинации кнопок в окне просмотра отчета.

```
function PEPrintControlsShowing (
    printJob: Word;
    var controlsShowing: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который отображен в окне просмотра, и требуется изменить комбинацию кнопок управления в этом окне
controlsShowing	Флаг отображения контрольных кнопок. Если TRUE, то кнопки отображены, если FALSE, то кнопки скрыты

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEPrintReport** — выполняет печать указанного отчета либо на принтер, либо вывод отчета в окно просмотра.

```
function PEPrintReport (
    reportFilePath: PChar;
```

```
    toDefaultPrinter: Bool;
    toWindow: Bool;
    title: PChar;
    left: Integer;
    top: Integer;
    width: Integer;
    height: Integer;
    style: Longint;
    parentWindow: HWnd
    ): Smallint stdcall;
```

Параметры функции	Описание параметров
`reportFilePath`	Строка, завершающаяся на NULL и содержащая пути и имя файла отчета, который необходимо распечатать
`toDefaultPrinter`	Флаг вывода на принтер по умолчанию
`toWindow`	Флаг вывода в окно просмотра
`title`	Строка, содержащая текст заголовка для окна просмотра отчета
`left`	Абсцисса верхнего левого угла окна просмотра отчета в пикселах
`top`	Ордината верхнего левого угла окна просмотра отчета в пикселах
`width`	Ширина окна просмотра отчета в пикселах
`height`	Высота окна просмотра отчета в пикселах
`style`	Стиль окна, которое будет создано. Формируется аналогично стилю, определяемому в функции PEOutputToWindow
`parentWindow`	Указатель на родительское окно, которое выступает носителем окна просмотра

Возвращаемые значения:

- PE_ERR_NOERROR — если отчет обработан без ошибок;
- код ошибки, если отчет обработан с ошибками.

Функция **`PEPrintWindow`** — вывод отчета в окно просмотра.

```
function PEPrintWindow (
    printJob: Word;
    waitUntilDone: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо вывести в окно просмотра
waitUntilDone	Этот параметр является зарезервированным и всегда должен быть TRUE

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEReimportSubreport** — переносит подотчет в указанный отчет.

```
function PEReimportSubreport (
    printJob: Word;
    subreportHandle: DWord;
    linkChanged: Bool;
    reimported: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который является основным
subreportHandle	Номер подотчета, который требуется перенести
linkChanged	Флаг, определяющий необходимость изменения взаимосвязей
reimported	Флаг, определяющий необходимость переноса подотчета

Возвращаемые значения:

- TRUE — если вызов функции произошел без ошибок и подотчет обновлен или перенесен с фиксацией изменений в связях;
- FALSE — если функция вызвана с ошибкой. Путь к подотчету неверен или перенос невозможен.

Замечание

Параметр linkChanged будет равен:

- FALSE, если подотчет перезагружен и связи зафиксированы;
- TRUE, если подотчет перезагружен, но связи неверны.

Параметр reimported будет равен:

- FALSE, если подотчет обновлен или перезагружен с неверным путем или с другой ошибкой;
- TRUE, если подотчет перезагружен с фиксацией связей либо с их ошибкой.

Функция **PESelectPrinter** — позволяет указать принтер, отличный от того, который определен в отчете или является принтером по умолчанию в Windows.

```
function PESelectPrinter (
    printJob: Word;
    driverName: PChar;
    printerName: PChar;
    portName: PChar;
    mode: PDeviceModeA
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета , для которого требуется сменить принтер
driverName	Строка, содержащая имя драйвера принтера, завершающаяся NULL
printerName	Строка, содержащая имя принтера, завершающаяся NULL
portName	Строка, которая содержит имя порта, используемого принтером, завершающаяся NULL
mode	Запись DEVMODE, которая относится к записям Windows API

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetAllowPromptDialog** — определяет необходимость отображения диалога, предлагающего ввести значения параметров в момент печати отчета.

```
function PESetAllowPromptDialog (
    printJob: Smallint;
    showPromptDialog: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета
`showPromptDialog`	Равен TRUE, когда изменение значений параметров возможно

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetAreaFormat** — позволяет установить параметры форматирования для выбранной секции отчета. Эта функция является аналогом функции PEGetAreaFormat, имеет тот же синтаксис и особенности применения.

```
function PESetAreaFormat (
    printJob: Word
    areaCode: Integer;
    options: PESectionOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, в который необходимо передать параметры форматирования
`areaCode`	Специфические коды разделов, для которых необходимо установить параметры форматирования, с помощью констант: `PE_ALLSECTIONS` `PE_SECT_PAGE_HEADER` `PE_SECT_PAGE_FOOTER` `PE_SECT_REPORT_HEADER` `PE_SECT_REPORT_FOOTER` `PE_SECT_GROUP_HEADER` `PE_SECT_GROUP_FOOTER` `PE_SECT_DETAIL`
`options`	Запись PESectionOptions, которая будет содержать информацию о параметрах форматирования выбранного раздела отчета

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetAreaFormatFormula** — позволяет изменить формулу форматирования выбранной секции отчета. Эта функция позволяет изменить текст только у существующей формулы. Добавить новую формулу с ее помощью невозможно.

```
function PESetAreaFormatFormula (
    printJob: Word;
    areaCode: Word;
    formulaName: Word;
    formulaString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо изменить формулу форматирования раздела
areaCode	Специфические коды разделов, для которых необходимо изменить параметры форматирования, задаются константами: PE_ALLSECTIONS PE_SECT_PAGE_HEADER PE_SECT_PAGE_FOOTER PE_SECT_REPORT_HEADER PE_SECT_REPORT_FOOTER PE_SECT_GROUP_HEADER PE_SECT_GROUP_FOOTER PE_SECT_DETAIL
formulaName	Константа, определяющая имя формулы форматирования, в которую вы хотите записать новую строку: PE_FFN_AREASECTION_VISIBILITY PE_FFN_SECTION_VISIBILITY PE_FFN_SHOW_AREA PE_FFN_NEW_PAGE_BEFORE PE_FFN_NEW_PAGE_AFTER PE_FFN_KEEP_TOGETHER PE_FFN_SUPPRESS_BLANK_SECTION PE_FFN_RESET_PAGE_N_AFTER PE_FFN_PRINT_AT_BOTTOM_OF_PAGE PE_FFN_UNDERLAY_SECTION PE_FFN_SECTION_BACK_COLOUR PE_FFN_SECTION_BACK_COLOR
formulaString	Строка, которая содержит текст формулы форматирования, завершающаяся значением NULL

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен;
- код ошибки PE_ERR_BADFORMULANAME, если формула не существует;
- код ошибки PE_ERR_BADFORMULATEXT, если есть ошибка в формуле.

Замечание

Эта функция вызывается перед PEStartPrintJob, иначе результат может быть неверным. Не все параметры могут быть использованы в любых разделах отчета.

Функция **PESetDialogParentWindow** — устанавливает ссылку на родительское окно, в котором открывается отчет.

```
function PESetDialogParentWindow (
    printJob: Word;
    parentWindow: HWnd
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо установить родительское окно
parentWindow	Указатель или ссылка на родительское окно

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetEventCallback** — определяет функцию обратной связи для выбранного отчета. Crystal Report Engine может сформировать событие, когда что-то произойдет внутри области действия Crystal Report Engine. CRPE вызовет функцию обратной связи и произведет уведомление о том, какого рода событие произошло. С помощью функции callbackProc пользователь может обработать событие по его идентификатору и выполнить надлежащие операции.

```
function PESetEventCallback (
    printJob: Word;
    callbackProc: Pointer
```

```
{Функция обратной связи должна быть представлена в виде:
Function callbacProc(eventID: Smallint;
param: Pointer;
userData: Pointer)}
): Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, для которого создается процедура обратной связи, обрабатывающая событие
`callbackProc`	Процедура обратной связи, которая будет обрабатывать события Crystal Report Engine. Она должна совпадать со стандартной процедурой обратной связи Windows. Описание методов создания процедур обратной связи представлено в Windows SDK
`userData`	Указатель на информацию, которую необходимо передать в процедуру обработки обратной связи. Это значение может быть равным 0

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Каждый отчет может иметь только одну функцию обратной связи.

Если `callbackProc` вернула TRUE, то действие по умолчанию для Crystal Report Engine произошло. Если `callbackProc` вернула FALSE, то действие, определенное по умолчанию, не было использовано. Пользователь должен обеспечить какое-либо поведение приложения. Для некоторых событий значение, возвращаемое `callbackProc`, игнорируется. Для уточнения списка событий, поддерживаемых CRPE, необходимо обратиться к документации.

Функция **`PESetFieldMappingType`** — устанавливает тип планировки полей для указанного отчета.

```
function PESetFieldMappingType (
    printJob: Smallint;
    var mappingType: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо изменить тип планировки полей
mappingType	Константа соответствующего типа PE_FM_XXX: PE_FM_AUTO_FLD_MAP PE_FM_CRPE_PROMPT_FLD_MAP PE_FM_EVENT_DEFINED_FLD_MAP

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetFont** — устанавливает шрифт для поля или текстового блока в указанной секции отчета.

```
function PESetFont (
    printJob: Word;
    sectionCode: Integer;
    scopeCode: Integer;
    faceName: PChar;
    fontFamily: Integer;
    fontPitch: Integer;
    charSet: Integer;
    pointSize: Integer;
    isItalic: Integer;
    isUnderlined: Integer;
    isStruckOut: Integer;
    weight: Integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить шрифт объекта
sectionCode	Код секции, для которой необходимо выполнить изменение шрифта. Используются константы, аналогичные константам, используемым в функциях PESetAreaFormat и PEGetAreaFormat
scopeCode	Константа, которая указывает, в каких элементах секции необходимо изменить шрифт (для того чтобы указать оба типа, необходимо использовать оператор OR): Константа PE_FIELDS

(продолжение)

Параметры функции	Описание параметров
scopeCode	Описание: устанавливает шрифт по умолчанию для всех полей в указанной секции отчета Константа PE_TEXT Описание: устанавливает шрифт по умолчанию для всех текстовых объектов, которые не были вставлены как поле, в указанной секции отчета
faceName	Имя шрифта, который необходимо использовать. Например Times New Roman. Установите 0, если не требуется изменять шрифт
fontFamily	Константа, определяющая семейство шрифтов: Константа FF_DONTCARE Описание: не менять Константа FF_ROMAN Описание: шрифт с переменной высотой — serif Константа FF_SWISS Описание: шрифт с фиксированной высотой — without serifs Константа FF_MODERN Описание: шрифт с фиксированной высотой — with or without serifs Константа FF_SCRIPT Описание: шрифт, подобный рукописному Константа FF_DECORATIVE Описание: декоративный шрифт
fontPitch	Константа, определяющая высоту шрифта: Константа DEFAULT_PITCH Значение — 0X00 Константа FIXED_PITCH Значение — 0X01 Константа VARIABLE_PITCH Значение — 0X02
charSet	Константа, определяющая тип кодировки, который будет использован: Константа ANSI_CHARSET Значение — 0 Константа DEFAULT_CHARSET Значение — 1

(продолжение)

Параметры функции	Описание параметров
charSet	Константа SYMBOL_CHARSET Значение — 2 Константа SHIFTJIS_CHARSET Значение — 128 Константа HANGEUL_CHARSET Значение — 129 Константа CHINESEBIG5_CHARSET Значение — 136 Константа OEM_CHARSET Значение — 255
pointSize	Определяет желаемую высоту шрифта. Установите 0, если не требуется вносить изменения
isItalic	TRUE для наклонного шрифта, FALSE для обычного и PE_UNCHANGED для того, чтобы использовать текущие установки
isUnderlined	TRUE для подчеркнутого шрифта, FALSE для обычного и PE_UNCHANGED для того, чтобы использовать текущие установки
isStruckOut	TRUE для перечеркнутого шрифта, FALSE для обычного и PE_UNCHANGED для того, чтобы использовать текущие установки
weight	Константа, определяющая толщину шрифта (установите 0, если не требуется вносить изменения): Константа FW_DONTCARE Значение — 0 Константа FW_THIN Значение — 100 Константа FW_EXTRALIGHT Значение — 200 Константа FW_LIGHT Значение — 300 Константа FW_NORMAL Значение — 400 Константа FW_MEDIUM Значение — 500 Константа FW_SEMIBOLD Значение — 600

(окончание)

Параметры функции	Описание параметров
	Константа FW_BOLD Значение — 700 Константа FW_EXTRABOLD Значение — 800 Константа FW_HEAVY Значение — 900 Константа FW_ULTRALIGHT Константа FW_EXTRALIGHT Константа FW_REGULAR Константа FW_NORMAL Константа FW_DEMIBOLD Константа FW_SEMIBOLD Константа FW_ULTRABOLD Константа FW_EXTRABOLD Константа FW_BLACK Константа FW_HEAVY

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetFormula** — позволяет изменить текст выбранной формулы. Эта функция изменяет только текст уже существующих в отчете формул. Вы не можете добавить новую формулу.

```
function PESetFormula (
    printJob: Word;
    formulaName: PChar;
    formulaString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо изменить текст формулы
formulaName	Строка, завершенная NULL, содержащая имя изменяемой формулы
formulaString	Строка, завершенная NULL, содержащая новый текст формулы

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен;
- код ошибки PE_ERR_BADFORMULANAME, если формула не существует;
- код ошибки PE_ERR_BADFORMULATEXT, если в формуле есть синтаксическая ошибка.

Функция **PESetFormulaSyntax** — устанавливается тип синтаксиса формул для использования при обработке отчета с помощью функций API.

```
procedure PESetFormulaSyntax (
    printJob: Word;
    var formulaSyntax: PEFormulaSyntax
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо изменить тип синтаксиса для формулы
formulaSyntax	Запись PEFormulaSyntax, которая будет содержать передаваемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGraphAxisInfo** — позволяет установить параметры осей у графика в отчете.

```
function PESetGraphAxisInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    var graphAxisInfo : PEGraphAxisInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется установить параметры осей у графика

(окончание)

Параметры функции	Описание параметров
sectionN	Номер секции, в которой находится график, требующий установки параметров для осей. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections
graphN	Номер графика, который необходимо настроить в указанной секции. Нумерация графиков начинается с 0 и основана на порядке вставки графика в отчет
graphAxisInfo	Запись PEGraphAxisInfo, которая содержит новую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGraphFontInfo** — позволяет установить шрифт для выбранного графика.

```
function PESetGraphFontInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    titleFontType : Word;
    var fontColourInfo : PEFontColorInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить шрифт для графика
sectionN	Номер секции, в которой находится график. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections
graphN	Номер графика, который необходимо настроить в указанной секции. Нумерация графиков начинается с 0 и основана на порядке вставки графика в отчет
titleFontType	Используются одна из констант: Константа PE_GTF_TITLEFONT Описание: шрифт основного заголовка графика Константа PE_GTF_SUBTITLEFONT Описание: шрифт подзаголовка

(окончание)

Параметры функции	Описание параметров
titleFontType	Константа PE_GTF_FOOTNOTEFONT Описание: шрифт ссылки Константа PE_GTF_GROUPSTITLEFONT Описание: шрифт заголовка группировки Константа PE_GTF_DATATITLEFONT Описание: шрифт заголовка таблицы данных Константа PE_GTF_LEGENDFONT Описание: шрифт легенды Константа PE_GTF_GROUPLABELSFONT Описание: шрифт ярлыка для группировки Константа PE_GTF_DATALABELSFONT Описание: шрифт ярлыка для таблицы данных
fontColourInfo	Запись PEFontColourInfo, которая будет содержать новую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGraphOptionInfo** — позволяет установить параметры отображения указанного графика.

```
function PESetGraphOptionInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    var graphOptionInfo : PEGraphOptionInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо установить параметры отображения графика
sectionN	Номер секции, в которой находится график. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections

(окончание)

Параметры функции	Описание параметров
graphN	Номер графика, который необходимо настроить в указанной секции. Нумерация графиков начинается с 0 и основана на порядке вставки графика в отчет
graphOptionInfo	Запись PEGraphOptionInfo, которая будет содержать новую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGraphTextDefaultOption** — позволяет включить или отключить текстовые заголовки графика, заданные по умолчанию.

```
function PESetGraphTextDefaultOption (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    titleType : Word;
    useDefault : Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо выполнить настройку графика
sectionN	Номер секции, в которой находится график. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections
graphN	Номер графика, который необходимо настроить в указанной секции. Нумерация графиков начинается с 0 и основана на порядке вставки графика в отчет
titleType	Тип заголовка. Используются константы: Константа PE_GTT_TITLE Описание: основной заголовок графика Константа PE_GTT_SUBTITLE Описание: подзаголовок графика Константа PE_GTT_FOOTNOTE Описание: сноска Константа PE_GTT_SERIESTITLE

(окончание)

Параметры функции	Описание параметров
titleType	Описание: заголовок для серий Константа PE_GTT_GROUPSTITLE Описание: заголовок для групп Константа PE_GTT_XAXISTITLE Описание: заголовок оси X Константа PE_GTT_YAXISTITLE Описание: заголовок оси Y Константа PE_GTT_ZAXISTITLE Описание: заголовок оси Z
useDefault	TRUE, если требуется выводить текстовые элементы графика, заданные по умолчанию; FALSE, если не требуется

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGraphTextInfo** — позволяет установить требуемую информацию в текстовые заголовки графика.

```
function PESetGraphTextInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    titleType : Word;
    title : PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется установить другие заголовки у выбранного графика
sectionN	Номер секции, в которой находится график. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections
graphN	Номер графика, который необходимо настроить в указанной секции. Нумерация графиков начинается с 0 и основана на порядке вставки графика в отчет

(окончание)

Параметры функции	Описание параметров
titleType	Тип заголовка. Используются константы: Константа PE_GTT_TITLE Описание: основной заголовок графика Константа PE_GTT_SUBTITLE Описание: подзаголовок графика Константа PE_GTT_FOOTNOTE Описание: сноска Константа PE_GTT_SERIESTITLE Описание: заголовок для серий Константа PE_GTT_GROUPSTITLE Описание: заголовок для групп Константа PE_GTT_XAXISTITLE Описание: заголовок оси "X" Константа PE_GTT_YAXISTITLE Описание: заголовок оси "Y" Константа PE_GTT_ZAXISTITLE Описание: заголовок оси "Z"
title	Строка, содержащая требуемый текст для указанного заголовка графика

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGraphTypeInfo** — позволяет установить тип указанного графика.

```
function PESetGraphTypeInfo (
    printJob : Word;
    sectionN : Smallint;
    graphN : Smallint;
    var graphTypeInfo : PEGraphTypeInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо установить тип графика

(окончание)

Параметры функции	Описание параметров
sectionN	Номер секции, в которой находится график. Этот параметр должен попадать в диапазон значений, возвращаемых функцией PEGetNSections
graphN	Номер графика, который необходимо настроить в указанной секции. Нумерация графиков начинается с 0 и основана на порядке вставки графика в отчет
graphTypeInfo	Запись PEGraphTypeInfo, которая будет содержать новую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGroupCondition** — позволяет изменить условия группировки.

```
function PESetGroupCondition (
    printJob: Word;
    sectionCode: Smallint;
    conditionField: PChar;
    condition: Smallint;
    sortDirection: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо изменить условия группировки
sectionCode	Код секции отчета, для которой необходимо задать условие группировки
conditionField	Имя поля, значение в котором необходимо изменить. Этот параметр можно получить с помощью функции PEGetHandleString, преобразовав значения conditionFieldHandle и conditionFieldLength, возвращаемые PEGetGroupCondition
condition	Условие группировки. Используется одна из констант PE_GC_XXX: Константа PE_GC_CONDITIONMASK Значение — 0x00FF Константа PE_GC_TYPEMASK Значение — 0x0F00

(продолжение)

Параметры функции	Описание параметров
`condition`	Константа PE_GC_TYPEOTHER Значение — 0x0000 Константа PE_GC_TYPEDATE Значение — 0x0200 Константа PE_GC_TYPEBOOLEAN Значение — 0x0400 Константа PE_GC_TYPETIME Значение — 0x0800 ***Все типы полей кроме Date и Boolean*** Константа PE_GC_ANYCHANGE Описание: отрабатывает группировку каждый раз при наличии изменений. ***Поля типа Date и DateTime*** Константа PE_GC_DAILY Описание: выполняется группировка по каждой дате Константа PE_GC_WEEKLY Описание: выполняется группировка по дате с учетом недели. Все даты, входящие в одну неделю, попадают в одну группу. Неделя начинается с воскресенья и завершается субботой Константа PE_GC_BIWEEKLY Описание: выполняется группировка по дате за двухнедельный период Константа PE_GC_SEMIMONTHLY Описание: выполняется группировка по дате за период в половину месяца Константа PE_GC_MONTHLY Описание: выполняется группировка по дате за период, равный месяцу Константа PE_GC_QUARTERLY Описание: выполняется группировка по дате за период, равный кварталу Константа PE_GC_SEMIANNUALLY Описание: выполняется группировка по дате за период, равный половине года Константа PE_GC_ANNUALLY Описание: выполняется группировка по дате за период, равный году

(продолжение)

Параметры функции	Описание параметров
condition	***Поля типа DateTime и Time Fields***
	Константа PE_GC_BYSECOND
	Описание: группировка по секундам
	Константа PE_GC_BYMINUTE
	Описание: группировка по минутам
	Константа PE_GC_BYHOUR
	Описание: группировка по часам
	Константа PE_GC_BYAMPM
	Описание: группировка по времени суток
	Поля типа Boolean
	Константа PE_GC_TOYES
	Описание: выполняется группировка по условию, когда значение поля, по которому выполняется сортировка и группировка, меняется с No на Yes
	Константа PE_GC_TONO
	Описание: выполняется группировка по условию, когда значение поля, по которому выполняется сортировка и группировка, меняется с Yes на No
	Константа PE_GC_EVERYYES
	Описание — выполняется группировка в том случае, когда значение поля, обеспечивающего сортировку и группировку, равно Yes
	Константа PE_GC_EVERYNO
	Описание: выполняется группировка в том случае, когда значение поля, обеспечивающего сортировку и группировку, равно No
	Константа PE_GC_NEXTISYES
	Описание: выполняется группировка в том случае, когда следующее значение поля, обеспечивающего сортировку и группировку, равно Yes
	Константа PE_GC_NEXTISNO
	Описание: выполняется группировка в том случае, когда следующее значение поля, обеспечивающего сортировку и группировку, равно No
sortDirection	Условие сортировки для выбранной секции группировки. Используются константы:
	Константа PE_SF_DESCENDING
	Описание: сортировка данных в порядке убывания (от Z к A, от 9 к 1)

(окончание)

Параметры функции	Описание параметров
	Константы PE_SF_ASCENDING Описание: сортировка данных в порядке возрастания (от A к Z, от 1 к 9) Константы PE_SF_ORIGINAL Описание: сортировка данных в том порядке, как они записаны в базу данных Константы PE_SF_SPECIFIED Описание: сортировка данных в заданном порядке с дополнительными условиями

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Не существует значений по умолчанию. Все значения для параметров должны быть заданы при использовании данной функции.

Если имеется формула, которая ссылается на поле суммирования, и вы изменяете условие на этом поле без фиксации самой формулы, то будет получена ошибка.

Эта функция вызывается всегда перед PEStartPrintJobor, иначе результат ее работы непредсказуем.

Функция **PESetGroupOptions** — позволяет изменить параметры группировки.

```
function PESetGroupOptions (
    printJob: Word;
    groupN: Smallint;
    var groupOptions: PEGroupOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить информацию о параметрах группировки
groupN	Номер группы. Нумерация начинается с нуля
groupOptions	Запись PEGroupOptions, содержащая параметры группировки

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetGroupSelectionFormula** — позволяет изменить формулу, накладывающую ограничения на сгруппированные в отчете данные. Формула может быть передана как параметр. Эта функция используется совместно с PEGetGroupSelectionFormula и PEGetHandleString.

```
function PESetGroupSelectionFormula (
    printJob: Word;
    formulaString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется установить новую формулу, ограничивающую сгруппированные значения в отчете
formulaString	Строка, завершающаяся NULL, с текстом формулы

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен в случае возникновения внутренней ошибки.

Функция **PESetMargins** — позволяет установить параметры границ отчета. Используется для установки специфических границ при печати отчета.

```
function PESetMargins (
    printJob: Word;
    left: Word;
    right: Word;
    top: Word;
    bottom: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо изменить параметры границ
left	Ширина левой границы
right	Ширина правой границы

(окончание)

Параметры функции	Описание параметров
top	Ширина верхней границы
bottom	Ширина нижней границы

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNDetailCopies** — позволяет распечатать множество копий детальной секции отчета. Для определения количества экземпляров детальной секции используется функция PEGetNDetailCopies.

```
function PESetNDetailCopies (
    printJob: Word;
    var nDetailCopies: Integer
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который передается информация о количестве распечатываемых секций Detail
nDetailCopies	Количество распечатываемых копий секции Detail

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthAlertConditionFormula** — позволяет задать условие для определенного предупреждения в отчете (Report Alert).

```
function PESetNthAlertConditionFormula (
    printJob : Word;
    alertN : Smallint;
    const formula : PChar
    ) : Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо задать условие на обработку предупреждения

(окончание)

Параметры функции	Описание параметров
`alertN`	Номер предупреждения, для которого задается условие
`formula`	Строка, содержащая текст условия, по которому обрабатывается предупреждение

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthAlertDefaultMessage** — позволяет определить текст сообщения по умолчанию для выбранного предупреждения в отчете.

```
function PESetNthAlertDefaultMessage (
    printJob : Word;
    alertN : Smallint;
    const text : PChar
    ) : Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, для которого требуется определить текст сообщения по умолчанию
`alertN`	Номер предупреждения, для которого требуется определить текст сообщения по умолчанию
`text`	Строка, содержащая текст сообщения по умолчанию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthAlertMessageFormula** — позволяет определить текст сообщения для предупреждения, связанного с отчетом.

```
function PESetNthAlertMessageFormula (
    printJob : Word;
    alertN : Smallint;
    const formula : PChar
    ) : Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, для которого требуется определить текст сообщения
`alertN`	Номер предупреждения, связанного с отчетом
`formula`	Строка, содержащая текст сообщения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **`PESetNthGroupSortField`** — позволяет установить параметры для одного из полей, по которому в отчете выполняется сортировка и группировка. Эта функция используется только для модификации уже существующего условия. Новое условие не может быть создано с ее помощью.

```
function PESetNthGroupSortField (
    printJob: Word;
    sortFieldN: Smallint;
    name: PChar;
    direction: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
`printJob`	Номер отчета, в котором необходимо изменить параметры поля, обеспечивающего группировку и сортировку данных
`sortFieldN`	Номер поля, которое необходимо перенастроить. Нумерация полей начинается с 0
`name`	Строка, содержащая наименование поля, обеспечивающего группировку и сортировку в отчете
`direction`	Порядок сортировки. Используются следующие константы: Константа PE_SF_DESCENDING Описание: сортировка данных в порядке убывания (от Z до A, от 9 до 1) Константа PE_SF_ASCENDING Описание: сортировка данных в порядке возрастания (от A до Z, от 1 до 9) Константа PE_SF_ORIGINAL Описание: только для полей группировки — сортировка данных в оригинальном порядке Константа PE_SF_SPECIFIED Описание: только для полей группировки — сортировка данных в специальном порядке. Только для чтения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthParameterDefaultValue** — позволяет установить значение по умолчанию в поле-параметр. Для уточнения информации о количестве значений по умолчанию используется функция PEGetNParameterDefaultValues.

```
function PESetNthParameterDefaultValue (
    printJob: Word;
    const parameterFieldName: PChar;
    const reportName: PChar;
    index: Smallint;
    var valueInfo: PEValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется установить значение по умолчанию для поля параметра
parameterFieldName	Строка, содержащая имя поля, которое является параметром отчета
reportName	Строка, содержащая имя отчета
index	Индекс значения по умолчанию, которое необходимо переопределить
valueInfo	Запись PEValueInfo, которая будет содержать значение по умолчанию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку "";
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PESetNthParameterField** — позволяет установить значение в указанное поле параметр отчета.

```
function PESetNthParameterField (
    printJob: Word;
    varN: Smallint;
    var varInfo: PEParameterFieldInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который содержит поле параметр, в котором необходимо изменить значение
parameterN	Номер параметра, который требуется изменить
parameterInfo	Запись PEParameterFieldInfo, которая используется для передачи требуемой информации

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthParameterValueDescription** — позволяет установить описание для поля параметра.

```
function PESetNthParameterValueDescription (
    printJob : Word;
    parameterFieldName : PChar;
    reportName : PChar;
    index : Smallint;
    valueDesc : PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется установить описание для поля параметра
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
index	Индекс параметра
valueDesc	Строка, содержащая текст, который будет использован в качестве описания для поля параметра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемые значения для параметра reportName:

- для основного отчета можно определить пустую строку ("");
- для подотчета необходимо указать путь к файлу и имя подотчета как NULL-завершенную строку.

Функция **PESetNthSortField** — позволяет указать поле отчета, по которому будет выполняться сортировка данных.

```
function PESetNthSortField (
    printJob: Word;
    sortFieldN: Smallint;
    name: PChar;
    var direction: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется установить специфические параметры сортировки
sortFieldN	Номер поля, которое будет использовано при сортировке. Нумерация полей начинается с 0
name	Строка, содержащая имя поля
direction	Константа, определяющая направление сортировки: Константа PE_SF_DESCENDING Описание: сортировка данных в порядке убывания (от Z до A, от 9 до 1) Константа PE_SF_ASCENDING Описание: сортировка данных в порядке возрастания (от A до Z, от 1 до 9) Константа PE_SF_ORIGINAL Описание: только для полей группировки — сортировка данных в оригинальном порядке Константа PE_SF_SPECIFIED Описание: только для полей группировки — сортировка данных в специальном порядке. Только для чтения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthTableLocation** — позволяет изменить местоположение указанной таблицы в выбранном отчете. Эта функция используется совместно с PEGetNthTableLocation.

```
function PESetNthTableLocation (
    printJob: Word;
    tableN: Integer;
    var location: PETableLocation
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо изменить местоположение таблиц, используемых для формирования набора данных для отчета
tableN	Номер таблицы, используемой при формировании отчета. Нумерация таблиц начинается с 0
location	Запись PETableLocation, содержащая данные о таблице. Формат строки зависит от типа источника, используемого для формирования отчета

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthTableLogOnInfo** — позволяет изменить параметры подключения к источнику данных для выбранной в отчете таблицы.

```
function PESetNthTableLogOnInfo (
    printJob: Word;
    tableN: Smallint;
    var logOnInfo: PELogOnInfo;
    propagateAcrossTables: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо изменить информацию о том, как подключаться к таблице в источнике данных

(окончание)

Параметры функции	Описание параметров
tableN	Номер таблицы, используемой для построения отчета. Нумерация таблиц начинается с 0
logOnInfo	Запись PELogOnInfo, содержащая всю необходимую для подключения к базе данных информацию
propagateAcrossTables	Если TRUE — программа выполнит подключение к базе данных с указанными параметрами для всех таблиц, которые присутствуют в отчете. Если FALSE — то указанные параметры будут применены только к выбранной таблице

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Для получения дополнительной информации смотрите описание функции PELogOnServer.

Функция **PESetNthTablePrivateInfo** — позволяет передать в отчет информацию, необходимую при использовании таких объектов данных, как ADO, DAO, RDO или CDO с соответствующими драйверами Active Data Drivers (crdb_ado.dll, crdb_dao.dll, crdb_odbc.dll, crdb_cdo.dll).

```
function PESetNthTablePrivateInfo (
    printJob: Word;
    tableN: Smallint;
    var privateInfo: PETablePrivateInfo
    ) :Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который использует таблицы, требующие информацию о сессии связи с источником
tableN	Номер таблицы, для которой необходимо передать информацию. Нумерация таблиц начинается с 0
privateInfo	Запись PETablePrivateInfo, содержащая необходимую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetNthTableSessionInfo** — позволяет передать информацию, необходимую для организации сессии с таблицами Microsoft Access, используемыми в отчете. Данная функция передает следующую информацию: User ID, Password и Session Handle.

```
function PESetNthTableSessionInfo (
    printJob: Word;
    tableN: Smallint;
    var sessionInfo: PESessionInfo;
    propagateAcrossTables: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, требующий передачи информации о сессии
tableN	Номер таблицы, используемой в отчете. Нумерация таблиц начинается с 0
sessionInfo	Запись PESessionInfo, которая содержит требуемую информацию
propagateAcrossTables	Если TRUE — программа выполнит подключение к базе данных с указанными параметрами для всех таблиц, которые присутствуют в отчете. Если FALSE — то указанные параметры будут применены только к выбранной таблице

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Эта функция может быть использована только для баз данных MS Access, которые требуют открытия сессии, прежде чем будет обеспечен доступ к базе.

Функция **PESetParameterMinMaxValue** — позволяет передать минимальное и максимальное значение в указанное поле параметр в отчете.

```
function PESetParameterMinMaxValue (
    printJob: Word;
    const parameterFieldName: PChar;
    const reportName: PChar;
    var valueMin: PEValueInfo;
    var valueMax: PEValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который необходимо передать минимальное и максимальное значение параметра
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
valueMin	Запись PEValueInfo, в которую возвращается информация о минимальном значении параметра
valueMax	Запись PEValueInfo, в которую возвращается информация о максимальном значении параметра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Предполагаемое значение параметра reportName:

- для основного отчета пустая строка ("");
- для подотчета пути и имени подотчета как строка, завершающаяся NULL.

Предполагаемые значения для параметров valueMin и valueMax:

- установите valueMin в NULL для передачи только максимального значения и установите значение, отличное от NULL, если valueMax равно NULL;
- установите valueMax в NULL для передачи только минимального значения и установите значение, отличное от ~NULL, если valueMin равно NULL.

Функция **PESetParameterPickListOption** — позволяет передать список настроек для параметра в указанном отчете.

```
function PESetParameterPickListOption (
    printJob : Word;
```

```
    parameterFieldName : PChar;
    reportName : PChar;
    var pickListOption : PEParameterPickListOption
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который необходимо передать список настроек для параметра
parameterFieldName	Строка, содержащая имя параметра
reportName	Строка, содержащая имя отчета
pickListOption	Запись PEParameterPickListOption, которая будет содержать передаваемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемое значение параметра reportName:

- для основного отчета пустая строка ("");
- для подотчета пути и имени подотчета как строка, завершающаяся NULL.

Функция **PESetParameterValueInfo** — позволяет передать информацию о значениях, которые могут быть записаны в поле параметр.

```
function PESetParameterValueInfo (
    printJob : Word;
    const parameterFieldName : PChar;
    const reportNam : PChar;
    var valueInfo : PEParameterValueInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в который необходимо передать информацию о параметре
parameterFieldName	Строка, содержащая имя поля параметра
reportName	Строка, содержащая имя отчета
valueInfo	Запись PEParameterValueInfo, в которой содержатся требуемые данные

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Ожидаемое значение параметра reportName:

- для основного отчета пустая строка ("");
- для подотчета пути и имени подотчета как строка, завершающаяся NULL.

Функция **PESetPrintDate** — позволяет изменить дату печати, которая определена в отчете.

```
function PESGetPrintDate (
    printJob: Word;
    var year: Word;
    var month: Word;
    var day: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить дату печати
year	Номер, соответствующий четырем цифрам года
month	Номер, соответствующий двум цифрам месяца
day	Номер, соответствующий двум цифрам числа

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Изменение значения даты печати может потребоваться в случае, когда вы печатаете отчет сегодня, а дата печати должна стоять другая.

Функция **PESetPrintOptions** — позволяет установить параметры для печати отчета.

```
function PESetPrintOptions (
    printJob: Word;
```

```
    var options: PEPrintOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменение текущих параметров печати
options	Запись PEPrintOptions, которая содержит требуемые параметры. Если этот параметр установить равным 0 (NULL), функция предоставит возможность пользователю задать требуемые параметры в диалоге настройки печати

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetReportOptions** — позволяет установить изменяемые параметры отчета, передаваемые с помощью записи PEReportAlertInfo.

```
function PESetReportOptions (
    printJob: Word;
    var reportOptions: PEReportOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить основные настройки
reportOptions	Запись PEReportAlertInfo, содержащая требуемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetReportSummaryInfo** — позволяет изменить суммарную информацию об отчете, которая была определена через диалог Summary Info, находящийся в Crystal Reports.

```
function PESetReportSummaryInfo (
    printJob: Word;
```

```
    var summaryInfo: PEReportSummaryInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета , в котором требуется изменить информацию
summaryInfo	Запись PEReportSummaryInfo, содержащая необходимую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetReportTitle** — используется для изменения заголовка отчета в разделе суммарной информации.

```
function PESetReportTitle (
    printJob: Word;
    title: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить заголовок
title	Строка, которая будет использована в качестве заголовка отчета

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetSectionFormat** — позволяет изменить параметры форматирования выбранной секции в отчете, которые передаются с помощью записи PESectionOptions.

```
function PESetSectionFormat (
    printJob: Word;
    sectionCode: Smallint;
    var options: PESectionOptions
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо изменить параметры форматирования указанной секции
sectionCode	Код секции, для которой требуется изменить параметры форматирования. Код секции можно получить с помощью функции PEGetSectionCode
options	Запись PESectionOptions, которая будет содержать требуемую информацию

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetSectionFormatFormula** — позволяет изменить формулу, с помощью которой выполняется форматирование указанной секции. Эта функция позволяет изменить только формулу, которая уже существует в отчете.

```
function PESetSectionFormatFormula (
    printJob: Word;
    sectionCode: Smallint;
    formulaName: Smallint;
    formulaString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется изменить текст формулы, по которой форматируется секция
sectionCode	Код секции, который может быть получен с помощью функции PEGetSectionCode
formulaName	Имя формулы, обеспечивающей форматирование. Используются константы: Константа PE_FFN_AREASECTION_VISIBILITY Описание: формула форматирования области отчета и отдельной секции Константа PE_FFN_SECTION_VISIBILITY Описание: формула форматирования секции Константа PE_FFN_SHOW_AREA Описание: формула форматирования области отчета Константа PE_FFN_NEW_PAGE_BEFORE Описание: формула форматирования области отчета и отдельной секции

(окончание)

Параметры функции	Описание параметров
`formulaName`	Константа PE_FFN_NEW_PAGE_AFTER Описание: формула форматирования области отчета и отдельной секции Константа PE_FFN_KEEP_TOGETHER Описание: формула форматирования области отчета и отдельной секции Константа PE_FFN_SUPPRESS_BLANK_SECTION Описание: формула форматирования секции Константа PE_FFN_RESET_PAGE_N_AFTER Описание: формула форматирования области отчета и отдельной секции Константа PE_FFN_PRINT_AT_BOTTOM_OF_PAGE Описание: формула форматирования области отчета и отдельной секции Константа PE_FFN_UNDERLAY_SECTION Описание: формула форматирования секции Константа PE_FFN_SECTION_BACK_COLOUR Описание: формула форматирования секции Константа PE_FFN_SECTION_BACK_COLOR Описание: формула форматирования секции
`formulaString`	Строка, содержащая новый текст формулы форматирования секции

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен;
- код ошибки PE_ERR_BADFORMULANAME — если формула не существует;
- код ошибки PE_ERR_BADFORMULATEXT — если в тексте формулы есть ошибка.

Функция **`PESetSectionHeight`** — изменяет высоту заданной секции в отчете.

```
function PESetSectionHeight (
    printJob: Word;
    sectionCode: Smallint;
    Height: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором изменяется информация о высоте секции
sectionCode	Код секции, который может быть получен с помощью функции PEGetSectionCode
height	Высота секции

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetSelectionFormula** — используется для изменения формулы, ограничивающей выборку данных в отчет из источника. Эта функция может быть использована как сама по себе, так и в составе с функциями PEGetSelectionFormula, PEGetHandleString.

```
function PESetSelectionFormula (
    printJob: Word;
    formulaString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется изменить формулу, ограничивающую выборку данных
formulaString	Строка, содержащая текст новой формулы

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — в случае возникновения внутренней ошибки.

Функция **PESetSQLExpression** — позволяет изменить SQL-выражение в указанном отчете.

```
function PESetSQLExpression (
    printJob: Word;
    const expressionName: PChar;
    const expressionString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором необходимо заменить SQL-выражение
expressionName	Строка, содержащая имя выражения
expressionString	Строка, содержащая новый текст SQL-выражения

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetSQLQuery** — используется для изменения SQL-запроса в отчете. Обычно используется для изменения части SQL-запроса, относящейся к предложению WHERE.

```
function PESetSQLQuery (
    printJob: Word;
    queryString: PChar
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется отредактировать SQL-запрос
queryString	Строка, содержащая измененный SQL-запрос

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetTrackCursorInfo** — позволяет установить параметры курсора, отображаемого в определенной области окна просмотра отчета. Эта функция может быть использована только при просмотре отчета.

```
function PESetTrackCursorInfo (
    printJob: Smallint;
    var cursorInfo: PETrackCursorInfo
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, в котором требуется управлять типом курсора
cursorInfo	Запись PETrackCursorInfo, которая будет содержать информацию о типе курсора

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PESetWindowOptions** — позволяет установить параметры окна просмотра отчета, включая доступные элементы контроля. Эта функция должна вызываться после PEOutputToWindow.

```
function PESetWindowOptions (
    printJob: Word;
    var options: PEWindowOptions
    ):Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который должен быть открыт в настраиваемом окне просмотра
PEWindowOptions	Запись PEWindowOptions, в которой будет содержаться информация о параметрах окна просмотра

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функции **PEShowFirstPage**, **PEShowLastPage**, **PEShowNextPage**, **PEShowPreviousPage**, **PEShowNthPage** — используются для вывода указанной страницы в окно просмотра. Для определения количества страниц в отчете используйте функцию PEGetNPages.

1. Показать первую страницу отчета.

```
function PEShowFirstPage (
    printJob: Word
    ): Bool stdcall;
```

2. Показать последнюю страницу отчета.

```
function PEShowLastPage (
    printJob: Word
    ): Bool stdcall;
```

3. Показать следующую страницу отчета.

```
function PEShowNextPage (
    printJob: Word
    ): Bool stdcall;
```

15 Зак. 1021

4. Показать предыдущую страницу отчета.

```
function PEShowPreviousPage (
    printJob: Word
    ): Bool stdcall;
```

5. Показать выбранную страницу отчета.

```
function PEShowNthPage (
    printJob: Word;
    pageN: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого необходимо выбирать страницы
Данный параметр появляется только в функции PEShowNthPage	
pageN	Номер страницы, которую требуется вывести в окно просмотра. Нумерация страниц начинается с 1

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEShowPrintControls** — позволяет выводить в окно просмотра отчета кнопки навигации — кнопки **First**, **Previous**, **Next** и **Last Page** и также кнопки **Cancel**, **Close**, **Export** и **Print to Printer**.

```
function PEShowPrintControls (
    printJob: Word;
    showPrintControls: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который будет выведен в окно просмотра
showPrintControls	TRUE — если кнопки должны быть показаны, FALSE — если кнопки не нужны

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEStartPrintJob** — выполняет печать отчета в окно просмотра, на принтер и т. п.

```
function PEStartPrintJob (
    printJob: Word;
    waitUntilDone: Bool
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который должен быть обработан
waitUntilDone	Этот параметр всегда должен быть TRUE. Он зарезервирован для следующих версий

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PETestNthTableConnectivity** — позволяет выполнить проверку параметров подключения к базе данных.

```
function PETestNthTableConnectivity (
    printJob: Word;
    tableN: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется проверить корректность параметров подключения к источнику данных
tableN	Номер таблицы, для которой требуется проверить параметры подключения. Нумерация таблиц в отчете начинается с 0

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Замечание

Если параметры подключения неверны, то можно отследить следующие ошибки:

Ошибка соединения	Код ошибки	Исправляется с помощью функции
Невозможно начать сессию	PE_ERR_DATABASESESSION	PESetNthTableSessionInfo
Невозможно подключиться к серверу	PE_ERR_DATABASELOGON	PESetNthTableLogOnInfo
Невозможно открыть таблицу	PE_ERR_DATABASELOCATION	PESetNthTableLocation

Функция **PEVerifyDatabase** — позволяет выполнить проверку структуры подключенной базы данных на соответствие структуре базы, определенной в отчете.

```
function PEVerifyDatabase
    printJob: Word
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, для которого требуется выполнить проверку

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Функция **PEZoomPreviewWindow** — позволяет управлять размерами просматриваемого отчета в определенных рамках: Full Page, Fit One Side, Fit Both Sides или другой вариант.

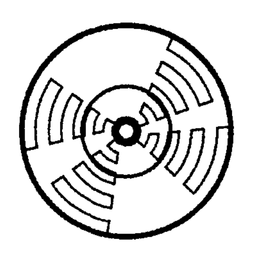
```
function PEZoomPreviewWindow (
    printJob: Word;
    level: Smallint
    ): Bool stdcall;
```

Параметры функции	Описание параметров
printJob	Номер отчета, который необходимо изменять в пределах окна просмотра
level	Варианты размеров, которые вы можете установить, должны попадать в пределы от 25% до 400%. Так же можно использовать некоторые константы: PE_ZOOM_FULL_SIZE PE_ZOOM_SIZE_FIT_ONE_SIDE PE_ZOOM_SIZE_FIT_BOTH_SIDES

Возвращаемые значения:

- TRUE — если вызов удачен;
- FALSE — если вызов неудачен.

Приложение 2



Список переменных, имеющих тип *ЗАПИСЬ* или *СТРУКТУРА*, используемых при работе с функциями Crystal Report Print Engine API

Переменная **PEAlertInstanceInfo** — содержит информацию о выбранном экземпляре Report Alert. Эта информация используется в функции PEGetNthAlertInstanceInfo.

```
type PEAlertInstanceInfo = record
    StructSize : Word;
    alertMessageLength : Smallint;
    alertMessage : HWnd;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEAlertInstanceInfo. Инициализация данного параметра осуществляется с помощью константы PE_SIZEOF_ALERT_INSTANCE_INFO
alsertMessageLength	Длина сообщения, создаваемого для данного экземпляра Report Alert
alertMessage	Указатель на строку, содержащую сообщение, создаваемое для данного экземпляра Report Alert

Переменная **PECloseButtonClickedEventInfo** — содержит информацию о событии, связанном с нажатием кнопки закрытия окна просмотра отчета. При нажатии кнопки закрытия окна просмотра отчета функция обратной связи будет вызываться с параметром EventId = PE_CLOSE_BUTTON_CLICKED_EVENT.

```
type PECloseButtonClickedEventInfo = record
    StructSize: Word;
    viewIndex: Word;
    windowHandle: HWnd;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PECloseButtonClickedEventInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_CLOSE_BUTTON_CLICKED_EVENT_INFO
viewIndex	Индекс, который указывает на закрываемое окно просмотра. Нумерация начинается с 0
windowHandle	Указатель на окно, в котором находится кнопка

Переменная **PEDrillOnDetailEventInfo** — содержит информацию, связанную с данными, возвращаемыми по параметру обратной связи Event Id = PE_DRILL_ON_DETAIL_EVENT.

```
type
PEFieldValueInfoDoublePtr = PEFieldValueInfoPtr
PEDrillOnDetailEventInfo = record
    StructSize: Word;
    selectedFieldIndex : Smallint;
    windowHandle : Longint;
    fieldValueList: PEFieldValueInfoDoublePtr;
    nFieldValue : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEDrillOnDetailEventInfo. Инициализация данного параметра осуществляется с помощью константы PE_SIZEOF_DRILL_ON_DETAIL_EVENT_INFO
selectedFieldIndex	Индекс, показывающий, какое поле drill-down было выбрано. Нумерация начинается с 0. Содержит 1, если поле не было выбрано
windowHandle	Указатель окна, в котором отработало событие drill-down
fieldValueList	Массив PEFieldValue. Память, выделенная под fieldValueList, освобождается после вызова функции обратной связи
nFieldValue	Индекс значения в списке fieldValueList. Нумерация начинается с 1

Переменная **PEDrillOnGroupEventInfo** — определяет информацию о специальной обработке группы по событию PE_DRILL_ON_GROUP_EVENT.

```
type
   PEPCharPointer = PChar
   PEDrillOnGroupEventInfo = record
   StructSize: Word;
   drillType: Word;
   windowHandle: HWnd;
   groupList: PEPCharPointer;
   groupLevel: Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEDrillOnGroupEventInfo. Инициализация данного параметра производится с помощью константы PE_SIZEOF_DRILL_ON_GROUP_EVENT_INFO
drillType	Тип специальной обработки. Используется одно из следующих значений: Константа PE_DE_ON_GROUP Описание компонентов — выделить сумму по группе или подгруппе Константа PE_DE_ON_GROUPTREE Описание — выделить дерево групп Константа PE_DE_ON_GRAPH Описание — выделить графические обьекты в группе Константа PE_DE_ON_MAP Описание — выделить регион на карте Константа PE_DE_ON_SUBREPORT Описание — выделить подотчет
windowHandle	Указатель окна, в котором произошло событие
groupList	Список имен групп в отчете и подотчетах. Память, которая занимается данным списком, освобождается после вызова функции обратной связи
groupLevel	Номер группы в списке имен

Переменная **PEEnableEventInfo** — определяет, какое событие группировки является доступным, а какое нет. Все события по умолчанию являются недоступными. Для изменения статуса события необходимо использовать функцию PEEnableEvent.

```
type PEEnableEventInfo = record
   StructSize: Word;
```

```
    startStopEvent: Smallint;
    readingRecordEvent: Smallint;
    printWindowButtonEvent: Smallint;
    drillEvent: Smallint;
    closePrintWindowEvent: Smallint;
    activatePrintWindowEvent: Smallint;
    fieldMappingEvent: Smallint;
    mouseClickEvent: Smallint;
    hyperlinkEvent: Smallint;
    launchSeagateAnalysisEvent: Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEEnableEventInfo. Инициализация данного значения выполняется с помощью константы PE_SIZEOF_ENABLE_EVENT_INFO
startStopEvent	Булево значение или PE_UNCHANGED. Событие Старт/Стоп
readingRecordEvent	Булево значение или PE_UNCHANGED. Событие чтения записи
printWindowButtonEvent	Булево значение или PE_UNCHANGED. Событие нажатия кнопки **Печать**
drillEvent	Булево значение или PE_UNCHANGED. Событие специальной выборки
closePrintWindowEvent	Булево значение или PE_UNCHANGED. Событие закрытия окна просмотра отчета
activatePrintWindowEvent	Булево значение или PE_UNCHANGED. Событие активизации окна просмотра отчета
fieldMappingEvent	Булево значение или PE_UNCHANGED. Событие связывания полей
mouseClickEvent	Булево значение или PE_UNCHANGED. Событие нажатия кнопки мыши
hyperlinkEvent	Булево значение или PE_UNCHANGED. Событие обработки гиперссылки
launchSeagateAnalysisEvent	Булево значение или PE_UNCHANGED. Событие связки с Seagate Analysis

Замечание

Каждый компонент записи может быть определен через значения TRUE, FALSE или PE_UNCHANGED.

Переменная **PEExportOptions** — содержит информацию о формате файла и способе обработки, которая получена с помощью функции PEGetExportOptions. Используется в функции PEExportTo при экспорте отчета.

```
type
   PEDllNameType = array[0..PE_DLL_NAME_LEN-1] or Char;
   PEExportOptions = record
   StructSize: Word;
   formatDLLName: PEDllNameType;
   formatType: Word;
   formatOptions: Pointer;
   destinationDLLName: PEDllNameType;
   destinationType: Word;
   destinationOptions: Pointer;
   nFormatOptionsBytes: Word;
   nDestinationOptionsBytes: Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEExportOptions. Инициализация данного значения выполняется с помощью константы PE_SIZEOF_EXPORT_OPTIONS
formatDLLName	Строка, содержащая имя библиотеки, длина строки PE_DLL_NAME_LEN = 64. Формат экспорта — Crystal Reports Format Используемая библиотека — u2fcr.dll Формат экспорта — Data Interchange Format Используемая библиотека — crxf_wordw.dll
formatType	Тип формата для экспорта отчета Crystal Reports. Выбирать можно из представленного списка: Формат экспорта — Crystal Reports Константа, определяющая тип, — UXFCrystalReportType Формат экспорта — Data Interchange Константа, определяющая тип, — UXFDIFType Формат экспорта — Word for Windows Константа, определяющая тип, — UXFWordWinType Формат экспорта — Word for DOS Константа, определяющая тип, — UXFWordDosType Формат экспорта — WordPerfect Константа, определяющая тип, — UXFWordPerfectType Формат экспорта — Quattro Pro 5.0 (WB1) Константа, определяющая тип, — UXFQP5Type

(продолжение)

Компоненты записи	Описание компонентов
`formatType`	Формат экспорта — Record Style (данные только из колонок) Константа, определяющая тип, — `UXFRecordType` Формат экспорта — Rich Text Format Константа, определяющая тип, — `UXFRichTextFormatType` Формат экспорта — Comma Separated Values (CSV) Константа, определяющая тип, — `UXFCommaSeparatedType` Формат экспорта — Tab Separated Values Константа, определяющая тип, — `UXFTabSeparatedType` Формат экспорта — Character Separated Values Константа, определяющая тип, — `UXFCharSeparatedType` Формат экспорта — Text Format (ASCII) Константа, определяющая тип, — `UXFTextType` Формат экспорта — Paginated Text (ASCII) Константа, определяющая тип, — `UXFPaginatedTextType` Формат экспорта — Tab Separated Text Константа, определяющая тип, — `UXFTabbedTextType` Формат экспорта — Lotus 1-2-3 (WK3) Константа, определяющая тип, — `UXFLotusWk3Type` Формат экспорта — Excel 4.0 Константа, определяющая тип, — `UXFXls4Type` Формат экспорта — Excel 5.0 Константа, определяющая тип, — `UXFXls5Type` Формат экспорта — Excel 5.0 Tabular Константа, определяющая тип, — `UXFXlsTypeTab` Формат экспорта — ODBC Константа, определяющая тип, — `UXFODBCType` Формат экспорта — HTML Константа, определяющая тип, — `UXFHTML3Type` Формат экспорта — Microsoft Internet Explorer 2 HTML Константа, определяющая тип, — `UXFExplorer2Type` Формат экспорта — Netscape 2 HTML Константа, определяющая тип, — `UXFNetscape2Type`

(продолжение)

Компоненты записи	Описание компонентов
`formatOptions`	Указатель на тип обработки информации типа дата или цифра при экспорте в некоторые виды форматов:  Это значение должно быть определено перед тем, как будет вызвана функция *PEStartPrintJob*. Формат экспорта — Data Interchange Константа, используемая для определения типа, — `UXFDIFOptions` Формат экспорта — Record Style (данные только из колонок) Константа, используемая для определения типа, — `UXFRecordStyleOptions` Формат экспорта — Comma Separated Values (CSV) Константа, используемая для определения типа, — `UXFCommaTabSeparatedOptions` Формат экспорта — Tab Separated Values Константа, используемая для определения типа, — `UXFCommaTabSeparatedOptions` Формат экспорта — Character Separated Values Константа, используемая для определения типа, — `UXFCharSeparatedOptions` Формат экспорта — Paginated Text Константа, используемая для определения типа, — `UXFPaginatedTextOptions` Формат экспорта — ODBC Константа, используемая для определения типа, — `UXFODBCOptions` Формат экспорта — HTML Константа, используемая для определения типа, — `UXFHTML3Options`
`destinationDLLName`	Указатель на строку, содержащую имя библиотеки, выполняющей требуемую обработку результата экспорта (длина строки определяется с помощью константы PE_DLL_NAME_LEN = 64): Вид экспорта — Disk File Используемая библиотека — `u2ddisk.dll` Вид экспорта — E-Mail (MAPI)

(продолжение)

Компоненты записи	Описание компонентов
`destinationDLLName`	Используемая библиотека — `u2dmapi.dll` Вид экспорта — E-Mail (VIM) Используемая библиотека — `u2dvim.dll` Вид экспорта — Microsoft Exchange Используемая библиотека — `u2dpost.dll`
`destinationType`	Тип обработки, поддерживаемый выбранной библиотекой. Вид экспорта — Disk File Константа, определяющая тип обработки, — `UXDDiskType` Вид экспорта — E-Mail (MAPI) Константа, определяющая тип обработки, — `UXDMAPIType` Вид экспорта — E-Mail (VIM) Константа, определяющая тип обработки, — `UXDVIMType` Вид экспорта — Microsoft Exchange Константа, определяющая тип обработки, — `UXDExchFolderType`
`destinationOptions`	Указатель на запись, содержащую информацию, используемую в PEExportOptions. **Предупреждение** Это значение должно быть определено прежде, чем будет вызвана функция *PEStartPrintJob*. Вид экспорта — Disk File Запись, используемая для определения требуемой информации, — `UXDDiskOptions` Вид экспорта — E-Mail (MAPI) Запись, используемая для определения требуемой информации, — `UXDMAPIOptions` Вид экспорта — E-Mail (VIM) Запись, используемая для определения требуемой информации, — `UXDVIMOptions` Вид экспорта — Microsoft Exchange Запись, используемая для определения требуемой информации, — `UXDPostFolderOptions`

(окончание)

Компоненты записи	Описание компонентов
nFormatOptionsBytes	Устанавливается с помощью PEGetExportOptions и игнорируется PEExportTo
nDestinationOptionsBytes	Устанавливается с помощью PEGetExportOptions и игнорируется PEExportTo

Замечание

Разработчики, использующие Visual Basic, должны строго следовать требованиям по работе со структурами.

Переменная **PEFieldMappingEventInfo** — содержит информацию, связанную со стыковкой полей отчета и полей базы данных.

```
type
    PEFieldMappingInfoPtr = PEReportFieldMappingInfo;
    PEFieldMappingInfoDoublePtr = PEFieldMappingInfoPtr;
    PEFieldMappingEventInfo = record
    StructSize : Word;
    reportFields : PEFieldMappingInfoDoublePtr;
    nReportFields : Word;
    databaseFields : PEFieldMappingInfoDoublePtr;
    nDatabaseFields : Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEFieldMappingEventInfo. Инициализация данного значения выполняется с помощью константы PE_SIZEOF_FIELDMAPPING_EVENT_INFO
reportFields	Указатель на массив PEReportFieldMappingInfo, содержащий информацию о полях в отчете
nReportFields	Размер массива reportFields, эквивалентен количеству полей в отчете
databaseFields	Указатель на массив PEReportFieldMappingInfo, содержащий информацию о полях в новой базе данных
nDatabaseFields	Размер массива databaseField

Переменная **PEFieldValueInfo** — определяет значение поля fieldValueList в записи PEDrillOnDetailEventInfo.

```
type
   PEFieldNameType = array[0..PE_FIELD_NAME_LEN-1] of Char;
   PEFieldValueInfo = record
   StructSize: Word;
   ignored: Word;
   fieldName: PEFieldNameType;
   fieldValue: PEValueInfo;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEFieldValueInfo. Инициализация данного значения выполняется с помощью константы PE_SIZEOF_FIELD_VALUE_INFO
ignored	Для выравнивания по 4 байта. Игнорируется
fieldName	Имя поля, длина строки определяется с помощью константы PE_FIELD_NAME_LEN = 512
fieldValue	Значение поля, указанного в fieldName

Переменная **PEFontColorInfo** — содержит информацию, относящуюся к заголовку графика.

```
type
   PEFaceNameType = array [0..PE_FACE_NAME_LEN-1] of Char;
   PEFontColorInfo = record
   StructSize: Word;
   faceName: PEFaceNameType;
   fontFamily: Smallint;
   fontPitch: Smallint;
   charSet: Smallint;
   pointSize: Smallint;
   isItalic: Smallint;
   isUnderlined: Smallint;
   isStruckOut: Smallint;
   weight: Smallint;
   color: COLORREF;
   twipSize: Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEFontColorInfo. Инициализация данного значения выполняется с помощью константы PE_SIZEOF_FONT_COLOR_INFO

(продолжение)

Компоненты записи	Описание компонентов
faceName	Массив наименований шрифтов, длина наименования определяется константой PE_FACE_NAME_LEN = 64. ***Пример*** "Times New Roman" Укажите пустую строку "", если не желаете изменять шрифт
fontFamily	Указатель на семейство шрифтов. Используются следующие константы: Константа FF_DONTCARE Описание — не требуются изменения Константа FF_ROMAN Описание — шрифт переменной высоты Константа FF_SWISS Описание — шрифт с постоянной высотой Константа FF_MODERN Описание — шрифт, имеющий современный стиль Константа FF_SCRIPT Описание — шрифт, имеющий рукописный стиль Константа FF_DECORATIVE Описание — декоративный шрифт
fontPitch	Высота шрифта. Используются константы, описанные в WINDOWS.H, или константа DEFAULT_PITCH для применения параметра по умолчанию
charSet	Тип кодировки. Используются константы, описанные в WINDOWS.H, или константа DEFAULT_CHARSET для применения параметра по умолчанию
pointSize	Определяет желаемый размер выбранного шрифта. Используется значение из этого поля либо из twipSize. Если оба элемента ненулевые, тогда данный параметр будет проигнорирован и используется twipSize. Установите 0 для twipSize и pointSize, если не желаете вносить изменения
isItalic	Определяет необходимость сделать шрифт наклонным. Если равен TRUE, то шрифт должен быть наклонным, если FALSE — прямым. Использование константы PE_UNCHANGED позволяет применить свойства, определенные по умолчанию

(окончание)

Компоненты записи	Описание компонентов
isUnderlined	Определяет необходимость сделать шрифт подчеркнутым. Если равен TRUE, то шрифт должен быть с подчеркиванием, если FALSE — не подчеркнутым. Использование константы PE_UNCHANGED позволяет применить свойства, определенные по умолчанию
isStruckOut	Определяет необходимость сделать шрифт перечеркнутым. Если равен TRUE, то шрифт должен быть перечеркнутым, если FALSE — не перечеркнутым. Использование константы PE_UNCHANGED позволяет применить свойства, определенные по умолчанию
weight	Вес шрифта. Используется константы, описанные в WINDOWS.H. Установите 0, если не требуется вносить изменения
color	Цвета RGB, описанные COLORREF, полное описание можно найти на http://msdn.microsoft.com/. Используйте PE_UNCHANGED_COLOR для применения параметров по умолчанию
twipSize	Повторное определение размера шрифта. Используется так же, как pointSize. Если оба элемента ненулевые, то параметр pointSize будет проигнорирован и используется текущее значение. Установите 0 для twipSize и pointSize, если не желаете вносить изменения

Переменная **PEFormulaSyntax** — используется в функциях PEGetFormulaSyntax и PESetFormulaSyntax для того, чтобы получить или передать параметры синтаксиса формулы, используемой при вызове через функции API.

```
type PEFormulaSyntax = record
    StructSize :Word;
    formulaSyntax :Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEFormulaSyntax. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_FORMULA_SYNTAX
formulaSyntax	Используйте PE_UNCHANGED, если не требуется вносить изменения, или константу, определяющую синтаксис формулы: Константа PE_FST_CRYSTAL

(окончание)

Компоненты записи	Описание компонентов
formulaSyntax	Описание — значение по умолчанию. Применяется в случае использования формулы без изменений
	Константа PE_FST_BASIC
	Описание — применяется в случае необходимости переопределения формулы

Переменная **PEGeneralPrintWindowEventInfo** — содержит информацию о событиях основного окна просмотра отчета. Эта запись может быть использована для обработки множества событий.

```
type PEGeneralPrintWindowEventInfo = record
    StructSize: Word;
    ignored: Word;
    windowHandle: HWnd;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEGeneralPrintWindowEventInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_GENERAL_PRINT_WINDOW_EVENT_INFO
ignored	Для выравнивания по 4 байта. Игнорируется
windowHandle	Указатель на окно просмотра отчета, в котором необходимо обработать событие

Переменная **PEGraphAxisInfo** — содержит информацию о сетке, диапазоне данных, форматах и параметрах осей указанного графика.

```
type PEGraphAxisInfo = record
    StructSize           : Word;
    groupAxisGridLine    : Smallint;
    dataAxisYGridLine    : Smallint;
    dataAxisY2GridLine   : Smallint;
    seriesAxisGridline   : Smallint;
    dataAxisYMinValue    : Double;
    dataAxisYMaxValue    : Double;
    dataAxisY2MinValue   : Double;
    dataAxisY2MaxValue   : Double;
    seriesAxisMinValue   : Double;
    seriesAxisMaxValue   : Double;
    dataAxisYNumberFormat  : Smallint;
    dataAxisY2NumberFormat : Smallint;
```

```
    seriesAxisNumberFormat : Smallint;
    dataAxisYAutoRange  : Smallint;
    dataAxisY2AutoRange : Smallint;
    seriesAxisAutoRange : Smallint;
    dataAxisYAutomaticDivision  : Smallint;
    dataAxisY2AutomaticDivision : Smallint;
    seriesAxisAutomaticDivision : Smallint;
    dataAxisYManualDivision  : Longint;
    dataAxisY2ManualDivision : Longint;
    seriesAxisManualDivision : LongInt;
    dataAxisYAutoScale : LongInt;
    dataAxisY2AutoScale : LongInt;
    seriesAxisAutoScale : LongInt;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEGraphAxisInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_GRAPH_AXIS_INFO
groupAxisGridLine	Параметры сетки. Используются константы PE_GGT_XXX или PE_UNCHANGED, если изменения не требуется
dataAxisYGridLine	Параметры сетки. Используются константы PE_GGT_XXX или PE_UNCHANGED, если изменения не требуется
dataAxisY2GridLine	Параметры сетки. Используются константы PE_GGT_XXX или PE_UNCHANGED, если изменения не требуется
seriesAxisGridline	Параметры сетки. Используются константы PE_GGT_XXX или PE_UNCHANGED, если изменения не требуется
dataAxisYMinValue	Минимальное значение для оси
dataAxisYMaxValue	Максимальное значение для оси
dataAxisY2MinValue	Минимальное значение для оси
dataAxisY2MaxValue	Максимальное значение для оси
seriesAxisMinValue	Минимальное значение для оси
seriesAxisMaxValue	Максимальное значение для оси
dataAxisYNumberFormat	Формат отображения цифровых значений на графике. Используются константы PE_GNF_XXX или PE_UNCHANGED, если изменения не требуется

(продолжение)

Компоненты записи	**Описание компонентов**
dataAxisY2NumberFormat	Формат отображения цифровых значений на графике. Используются константы PE_GNF_XXX или PE_UNCHANGED, если изменения не требуется
seriesAxisNumberFormat	Формат отображения цифровых значений на графике. Используются константы PE_GNF_XXX или PE_UNCHANGED, если изменения не требуется
dataAxisYAutoRange	Если TRUE, тогда диапазон оси определяется автоматически, или PE_UNCHANGED, если изменять параметры не требуется
dataAxisY2AutoRange	Если TRUE тогда диапазон оси определяется автоматически, или PE_UNCHANGED, если изменять параметры не требуется
seriesAxisAutoRange	Если TRUE тогда диапазон оси определяется автоматически или PE_UNCHANGED, если изменять параметры не требуется
dataAxisYAutomaticDivision	PE_ADM_AUTOMATIC, PE_ADM_MANUAL или PE_UNCHANGED, если изменять параметры не требуется
dataAxisY2AutomaticDivision	PE_ADM_AUTOMATIC, PE_ADM_MANUAL или PE_UNCHANGED, если изменять параметры не требуется
seriesAxisAutomaticDivision	PE_ADM_AUTOMATIC, PE_ADM_MANUAL или PE_UNCHANGED, если изменять параметры не требуется
dataAxisYManualDivision	Если dataAxisYAutomaticDivision равен PE_ADM_AUTOMATIC, то данное поле игнорируется
dataAxisY2ManualDivision	Если dataAxisY2AutomaticDivision равен PE_ADM_AUTOMATIC, то данное поле игнорируется
seriesAxisManualDivision	Если seriesAxisAutomaticDivision равен PE_ADM_AUTOMATIC, то данное поле игнорируется
dataAxisYAutoScale	Если TRUE, тогда ось будет иметь автоматически устанавливаемую шкалу, или PE_UNCHANGED, если изменять параметры не требуется

(окончание)

Компоненты записи	Описание компонентов
dataAxisY2AutoScale	Если TRUE, тогда ось будет иметь автоматически устанавливаемую шкалу, или PE_UNCHANGED, если изменять параметры не требуется
seriesAxisAutoScale	Если TRUE, тогда ось будет иметь автоматически устанавливаемую шкалу, или PE_UNCHANGED, если изменять параметры не требуется

Переменная **PEGraphOptionInfo** — содержит информацию о виде графика и используется в функциях PEGetGraphOptionInfo и PESetGraphOptionInfo.

```
type PEGraphOptionInfo = record
    StructSize        : Word;
    graphColour       : Smallint;
    showLegend        : Smallint;
    legendPosition    : Smallint;
    pieSize           : Smallint;
    detachedPieSlice  : Smallint;
    barSize           : Smallint;
    verticalBars      : Smallint;
    markerSize        : Smallint;
    markerShape       : Smallint;
    dataPoints        : Smallint;
    dataValueNumberFormat : Smallint;
    viewingAngle      : Smallint;
    legendLayout      : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEGraphOptionsInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_GRAPH_OPTION_INFO
graphColour	Используется одна из констант PE_GCR_XXX или PE_UNCHANGED, если изменять параметры не требуется
showLegend	Если TRUE, то требуется показывать легенду к графику, если FALSE, то легенду необходимо скрыть или PE_UNCHANGED, если изменять параметры не требуется
legendPosition	Используется одна из констант PE_GLP_XXX или PE_UNCHANGED, если изменять параметры не требуется. Влияет только в случае, когда showLegend равно TRUE

(окончание)

Компоненты записи	Описание компонентов
pieSize	Для графиков типа PIE и DOUGHNUT используется одна из констант PE_GPS_XXX или PE_UNCHANGED, если изменять параметры не требуется
detachedPieSlice	Для графиков типа PIE и DOUGHNUT используется одна из констант PE_GDPS_XXX или PE_UNCHANGED, если изменять параметры не требуется
barSize	Для графиков типа BAR используется одна из констант PE_GBS_XXX или PE_UNCHANGED, если изменять параметры не требуется
verticalBars	Для графиков типа BAR равна TRUE, если график должен иметь вертикальные области; равна FALSE, если вертикальных областей не будет; равна PE_UNCHANGED, если изменять параметры графика не требуется
markerSize	Для линейных графиков и графиков типа BAR используется одна из констант PE_GMS_XXX или PE_UNCHANGED, если изменять параметры графика не требуется
markerShape	Для линейных графиков и типа BAR используется одна из констант PE_GMSP_XXX или PE_UNCHANGED, если изменять параметры графика не требуется
dataPoints	Используется одна из констант PE_GDP_XXX или PE_UNCHANGED, если изменять параметры графика не требуется
dataValueNumberFormat	Используется одна из констант PE_GNF_XXX или PE_UNCHANGED, если изменять параметры графика не требуется
viewingAngle	Для трехмерных графиков используется одна из констант PE_GVA_XXX или PE_UNCHANGED, если изменять параметры графика не требуется
legendLayout	Определяет вид легенды. Используется одна из констант PE_GLL_XXX

Переменная **PEGraphTypeInfo** — содержит информацию о типе графика. Используется в функциях PEGetGraphTypeInfo и PESetGraphTypeInfo.

```
type PEGraphTypeInfo = record
    StructSize   : Word;
    graphType    : Smallint;
    graphSubtype : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEGraphTypeInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_GRAPH_TYPE_INFO
graphType	Используется одна из констант PE_GT_XXX или PE_UNCHANGED, если изменять параметры не требуется
graphSubtype	Используется одна из констант PE_GST_XXX или PE_UNCHANGED, если изменять параметры не требуется

Переменная **PEGroupOptions** — содержит информацию о параметрах группировки. Эта информация используется в функции PEGetGroupOptions, для того чтобы получить текущую информацию, и в функции PESetGroupOptions, чтобы передать новые параметры.

```
type
   PEFieldNameType = array[0..PE_FIELD_NAME_LEN-1] of Char;
   PEGroupOptions = record
   StructSize: Word;
   condition: Smallint;
   fieldName: PEFieldNameType;
   sortDirection: Smallint;
   repeatGroupHeader: Smallint;
   keepgroupTogether: Smallint;
   topOrBottomNGroups: Smallint;
   topOrBottomNSortFieldName: PEFieldNameType;
   nTopOrBottomGroups: Smallint;
   discardOtherGroups: Smallint
   hierarchicalSorting: Integer
   instanceIDField: String
   parentIDField: String
   groupIndent: Longint
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEGroupOptions. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_GROUP_OPTIONS
Condition	Условия для выбранной секции группировки. Когда необходимо получить информацию о параметрах, используются константы PE_GC_TYPEMASK и PE_GC_CONDITIONMASK. Когда требуется передать параметры группировки, используется одна из констант PE_GC_XXX, и используется PE_UNCHANGED, если не требуется вносить изменения в условия группировки отчета

(продолжение)

Компоненты записи	Описание компонентов
Fieldname	Имя поля базы данных, которое используется в качестве условия группирования отчета, длина строки определяется константой PE_FIELD_NAME_LEN = 512
sortDirection	Константа, указывающая на порядок сортировки, описанная ранее, или PE_UNCHANGED, если не требуется вносить изменений
repeatGroupHeader	TRUE, FALSE или PE_UNCHANGED, если не требуется вносить изменений
keepGoupTogether	TRUE, FALSE или PE_UNCHANGED, если не требуется вносить изменений
topOrBottomNGroups	Используется одна из представленных ниже констант PE_GO_TBN_XXX или PE_UNCHANGED, если не требуется вносить изменений: Константа PE_GO_TBN_ALL_GROUPS_UNSORTED Описание — нет сортировки по группе или Top/Bottom N сортировки для данного уровня группировки Константа PE_GO_TBN_ALL_GROUPS_SORTED Описание — данная константа отвечает за группировку и сортировку Top/Bottom N Константа PE_GO_TBN_TOP_N_GROUPS Описание — выбрана группа с сортировкой Top N Константа PE_GO_TBN_BOTTOM_N_GROUPS Описание — выбрана группа с сортировкой Bottom N
topOrBottomNSortFieldName	Имя поля, которое используется для группировки и специальной сортировки. Длина имени определяется константой PE_FIELD_NAME_LEN = 512. Можно использовать синтаксическую конструкцию формулы для изменения данного параметра
nTopOrBottomGroups	Количество групп, которое должно быть выведено по условию Top N/ Bottom N. Используйте 0 для того, чтобы выполнить сортировку данных по группам без дополнительных условий, и PE_UNCHANGED, если требуется оставить отчет без изменений
discardOtherGroups	Указатель на необходимость скрытия данных, которые попадают в раздел Others, или PE_UNCHANGED, если требуется использовать настройки отчета по умолчанию

(окончание)

Компоненты записи	Описание компонентов
hierarchicalSorting	Указывает на необходимость иерархической сортировки отчета или PE_UNCHANGED в случае необходимости использования параметров, заданных заранее
instanceIDField	Указатель на экземпляр поля, используемый для иерархической группировки и сортировки. Длина идентификатора определяется константой PE_FIELD_NAME_LEN = 512
parentIDField	Указатель на поле, которое используется в качестве родителя при иерархической группировке и сортировке. Длина имени определяется константой PE_FIELD_NAME_LEN = 512
groupIndent	Повторный идентификатор, указывающий на иерархическую группировку и сортировку

Переменная **PEGroupTreeButtonClickedEventInfo** — представляет информацию о событии нажатия кнопки **Показать дерево групп (Group Tree)**, полученное при обработке функции обратной связи, которая вызывается по событию с идентификатором PE_GROUP_TREE_BUTTON_CLICKED_EVENT.

```
type PEGroupTreeButtonClickedEventInfo = record
    StructSize: Word;
    visible: Smallint;
    windowHandle: HWnd;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEGroupTreeButtonClickedEventInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_GROUP_TREE_BUTTON_CLICKED_EVENT_INFO
visible	Показывает, когда дерево групп скрыто, а когда видно
windowHandle	Указатель на окно просмотра отчета, в котором произошло событие

Переменная **PEJobInfo** — содержит информацию о процессе обработки отчета, находящемся в активном состоянии. Эта информация используется в функции PEGetJobStatus.

```
type PEJobInfo = record
    StructSize: Word;
```

```
    NumRecordsSelected: Longint;
    NumRecordsPrinted: Longint;
    DisplayPageN: Word;
    LatestPageN: Word;
    StartPageN: Word;
    PrintEnded: Bool;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEJobInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_JOB_INFO
NumRecordsRead	Номер записи, которая в данный момент находится в режиме чтения
NumRecordsSelected	Номер записи, выбранной для включения в отчет из всего множества записей, считанных из источника
NumRecordsPrinted	Количество записей, реально распечатанных или показанных
DisplayPageN	Номер страницы, показанной на текущий момент в окне просмотра отчета
LatestPageN	Показывает номер страницы, которая была сформирована в момент просмотра. В случае завершения печати этот номер будет совпадать с номером последней страницы отчета
StartPageN	Номер начальной страницы отчета. В нормальной ситуации номер должен быть равен 1, но можно определить другое значение с помощью функции PESetPrintOptions
printEnded	Показывает завершение процесса печати. • TRUE показывает, что процесс завершен. • FALSE показывает, что процесс не закончился. Если "печать" подразумевает вывод отчета для просмотра, то True будет получено только тогда, когда будет прочитана последняя страница

Переменная **PELogOnInfo** — содержит информацию, используемую для подключения к источнику данных и построению отчета. Эта информация используется в функциях PEGetNthTableLogOnInfo, PESetNthTableLogOnInfo, PELogOnServer и PELogOffServer для подключения и отключения от источников данных.

```
type
    PELogonServerType = array[0..PE_SERVERNAME_LEN-1] of Char;
```

```
    PELogonDBType = array[0..PE_DATABASENAME_LEN-1] of Char;
    PELogonUserType = array[0..PE_USERID_LEN-1] of Char;
    PELogonPassType = array[0..PE_PASSWORD_LEN-1] of Char;
    PELogOnInfo = record
    StructSize: Word;
    ServerName: PELogonServerType;
    DatabaseName: PELogonDbType;
    UserId: PELogonUserType;
    Password: PELogonPassType;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PELogOnInfo. Инициализация производится с помощью константы PE_SIZEOF_LOGON_INFO
ServerName	Строка, содержащая имя сервера базы данных, длина строки определяется константой PE_SERVERNAME_LEN = 128
DatabaseName	Строка, содержащая имя базы данных, длина строки определяется константой PE_DATABASENAME_LEN = 128
UserID	Имя пользователя базы данных, длина строки определяется константой PE_USERID_LEN = 128
Password	Пароль пользователя, длина строки определяется константой PE_PASSWORD_LEN = 128

Переменная **PEMouseClickEventInfo** — содержит информацию, связанную с событием нажатия кнопки мыши. Обрабатывается при вызове функции обратной связи, когда идентификатор события ID равен PE_RIGHT/MIDDLE/LEFT_CLICK_EVENT.

```
type PEMouseClickEventInfo = record
    StructSize : Word;
    windowHandle : LongInt;
    clickAction : Integer;
    clickFlags : Integer;
    xOffset : Integer;
    yOffset : Integer;
    fieldValue : PEValueInfo;
    objectHandle : DWord;
    sectionCode : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEMouseClickEventInfo. Инициализируется с помощью константы PE_SIZEOF_MOUSE_CLICK_EVENT_INFO

(продолжение)

Компоненты записи	Описание компонентов
windowHandle	Указатель на окно просмотра отчета, в котором произошло событие нажатия кнопки мыши
clickAction	Индикатор действия. Используется одна из представленных ниже констант PE_MOUSE_XXX: Константа PE_MOUSE_NOTSUPPORTED Описание — события, связанные с мышью, не поддерживаются Константа PE_MOUSE_DOWN Описание — кнопка мыши нажата Константа PE_MOUSE_UP Описание — кнопка мыши отпущена Константа PE_MOUSE_DOUBLE_CLICK Описание — только в случае двойного щелчка на левой кнопке мыши или на центральной кнопке для трехкнопочной мыши
clickFlags	Указывает на источник события. Возможны комбинации из констант, представленных ниже: Константа PE_CF_NONE Значение — 0x0000 Описание — кнопки не нажаты Константа PE_CF_LBUTTON Значение — 0x0001 Описание — левая кнопка Константа PE_CF_RBUTTON Значение — 0x0002 Описание — правая кнопка Константа PE_CF_SHIFTKEY Значение — 0x0004 Описание — колесо мыши, если имеется Константа PE_CF_CONTROLKEY Значение — 0x0008 Описание — контрольная кнопка, если имеется Константа PE_CF_MBUTTON Значение — 0x00010 Описание — Центральная кнопка, если имеется
Xoffset	Координаты курсора мыши по оси X в момент нажатия кнопки, в пикселах

(окончание)

Компоненты записи	Описание компонентов
Yoffset	Координаты курсора мыши по оси Y в момент нажатия кнопки, в пикселах
fieldValue	Запись PEValueInfo, содержащая информацию о значениях, определенных для объекта, на котором щелкнули курсором мыши, исключая поля типа MEMO и BLOB, в противном случае элемент valueType = PE_VI_NOVALUE
objectHandle	Указатель на обрабатываемый объект
sectionCode	Код секции, в которой находится обрабатываемый объект

Переменная **PEParameterFieldInfo** — содержит информацию, связанную с полем параметром в отчете. Используется в функции PEGetNthParameterField для того, чтобы получить информацию о поле, и в функции PESetNthParameterField для того, чтобы изменить указанное поле параметр.

```
type
   PEParameterFieldValueType = array[0..PE_PF_NAME_LEN-1] of Char;
   PE_PF_ReportNameType = array[0..PE_PF_REPORT_NAME_LEN-1] of Char;
   PEParameterFieldNameType  = array [0..PE_PF_NAME_LEN-1] of Char;
   PEParameterFieldEditMaskType = array [0..PE_PF_EDITMASK_LEN-1]
   of Char;
   PEParameterFieldInfo = record
   structSize: Word;
   ValueType: Word;
   DefaultValueSet: Word;
   CurrentValueSet: Word;
   Name: PEParameterFieldNameType;
   Prompt: PEParameterFieldTextType;
   DefaultValue: PEParameterFieldValueType;
   CurrentValue: PEParameterFieldValueType;
   ReportName: PE_PF_ReportNameType;
   needsCurrentvalue: Word;
   isLimited: Word;
   MinSize: Double;
   MaxSize: Double;
   EditMask: PEParameterFieldEditMaskType;
   isHidden: Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEParameterFieldInfo. Инициализация данного параметра производится с помощью константы PE_SIZEOF_PARAMETER_FIELD_INFO

(окончание)

Компоненты записи	Описание компонентов
ValueType	Тип данных, используемый у поля-параметра. В Crystal Report поддерживаются типы данных, ассоциированные с константами PE_PF_XXX
DefaultValueSet	Если TRUE, то требуется установить новое значение; если FALSE, то значение по умолчанию остается неизменным
CurrentValueSet	Если TRUE, то новое значение в поле параметр было установлено, если FALSE, то значение не менялось
Name	Имя поля-параметра. Длина имени соответствует константе PE_PF_NAME_LEN = 256
Prompt	Текст подсказки для поля параметра. Длина текста PE_PF_PROMPT_LEN = 256
DefaultValue	Если DefaultValueSet равно FALSE, то данное значение не используется. DefaultValue может быть по типу Number, Currency, Date, DateTime, Time, Boolean или String. Длина определяется константой PE_PF_VALUE_LEN = 256
CurrentValue	Если CurrentValueSet равно FALSE, данное значение не используется. CurrentValue может быть по типу Number, Currency, Date, DateTime, Time, Boolean или String. Длина определяется константой PE_PF_VALUE_LEN = 256
ReportName	Имя отчета. Длина имени определяется константой PE_PF_REPORT_NAME_LEN = 128
needsCurrentValue	Возвращает FALSE, если параметр является связанным, не используется или использует текущее значение
isLimited	Если TRUE, то: • для строковых типов указывает на ограничение длины; • для других типов ограничение диапазона
MinSize	Минимальная длина строки или минимальная цифра
MaxSize	Максимальная длина строки или максимальная цифра
EditMask	Маска, длина шаблона определяется константой PEP_PF_EDITMASK_LEN = 256. Поддерживается только для строковых параметров
isHidden	TRUE используется только при наличии внутренних переменных

Замечание

Если вы желаете присвоить параметру значение NULL, выберите в качестве передаваемого значения CRWNULL.

Переменная **PEParameterPickListOption** — содержит информацию, связанную с методом сортировки по параметру.

```
type PEParameterPickListOption = record
    StructSize       : Word;
    showDescOnly     : Smallint;
    sortMethod       : Smallint;
    sortBasedOnDesc  : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEParameterFieldInfo. Инициализация данного параметра осуществляется с помощью константы PE_SIZEOF_PICK_LIST_OPTION
showDescOnly	TRUE/FALSE или PE_UNCHANGED, если все остается без изменений
sortMethod	Одна из констант PE_OR_XXX или PE_UNCHANGED в случае отсутствия изменений
sortBasedOnDesc	TRUE/FALSE или PE_UNCHANGED, если все остается без изменений

Переменная **PEParameterValueInfo** — содержит информацию о типе значения, которое используется в поле параметре.

```
type PEParameterValueInfo = record
    StructSize : Word;
    isNullable : Smallint;
    disallowEditing : Smallint;
    allowMultipleValues : Smallint;
    hasDiscreteValues : Smallint;
    partOfGroup : Smallint;
    groupNum : Smallint;
    mutuallyExclusiveGroup : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEParameterValueInfo. Установка данного параметра осуществляется через константу PE_SIZEOF_PARAMETER_VALUE_INFO

(окончание)

Компоненты записи	Описание компонентов
isNullable	Разрешено или нет использование в качестве значения параметра NULL и равно TRUE/FALSE или PE_UNCHANGED, если не требуется ничего менять
disallowEditing	Указатель на возможность редактирования поля-параметра. Равно TRUE/FALSE или PE_UNCHANGED, если все остается без изменений
allowMultipleValues	Указатель на допустимость наличия множественных значений. Равно TRUE/FALSE или PE_UNCHANGED, если все остается без изменений
hasDiscreteValues	Указатель на свойства поля параметра: Константа PE_DR_HASRANGE Описание — допустимо использование только диапазона Константа PE_DR_HASDISCRETE Описание — допустимо использование только дискретных значений Константа PE_DR_HASDISCRETEANDRANGE Описание — допустимо использование любых вариантов
partOfGroup	Указатель на то, что параметр является частью условия группировки. Равно TRUE/FALSE или PE_UNCHANGED, если изменения не нужны
groupNum	Номер группы или PE_UNCHANGED, если все определено по умолчанию
mutuallyExclusiveGroup	Указатель на то, что параметр является условием для взаимоисключающих групп. Равно TRUE/FALSE или PE_UNCHANGED, если условие не обязательно

Переменная **PEPrintOptions** — содержит спецификацию печати, которая используется в PEGetPrintOptions для получения текущих настроек и PESetPrintOptions для определения новых.

```
type
   PEOutputFileNameType = array [0..PE_FILE_PATH_LEN-1] of Char;
   PEPrintOptions  = record
   StructSize: Word;
   StartPageN: Word;
   StopPageN: Word;
   nReportCopies: Word;
   Collation: Word;
   outputFileName: PEOutputFileNameType ;
end;
```

16 Зак. 1021

Компоненты записи	Описание компонентов
StructSize	Размер записи PEPrintOptions. Определяется через константу PE_SIZEOF_PRINT_OPTIONS
startPageN	Указатель номера первой страницы для печати
stopPageN	Указатель номера последней страницы при печати
nReportCopies	Количество копий, которое требуется распечатать
collation	Индикатор, указывающий на необходимость разбора копий при печати Константа PE_UNCOLLATED Описание — печать множества экземпляров без разбора. Пример порядка страниц = 1, 1, 1, 2, 2, 2, 3, 3, 3 и т. д. Константа PE_COLLATED Описание — печать множества экземпляров, с разбором на папки. Пример порядка страниц = 1, 2, 3,..., 1, 2, 3, ... и т. д. Константа PE_DEFAULTCOLLATION Описание — печать отчета с теми параметрами, которые заданы заранее при формировании отчета
outputFileName	Указатель пути и имени файла, если отчет печатается в файл. Длина всего пути должна соответствовать значению, определенному константой PE_FILE_PATH_LEN = 512

Переменная **PEReadingRecordsEventInfo** — содержит информацию о читаемой записи. Считывается в тот момент, когда вызывается функция обратной связи по событию с идентификатором ID = PE_READING_RECORDS_EVENT.

```
type PEReadingRecordsEventInfo = record
    StructSize: Word;
    cancelled: Smallint;
    recordsRead: Longint;
    recordsSelected: Longint;
    done: Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEReadingRecordsEventInfo. Определяется с помощью константы PE_SIZEOF_READING_RECORDS_EVENT_INFO
cancelled	Возвращает FALSE, если чтение записей было отменено
recordsRead	Количество считанных записей
recordsSelected	Количество выбранных записей
done	Флаг завершения считывания записей

Переменная **PEReportFieldMappingInfo** — содержит информацию о взаимосвязи полей отчета с полями базы данных.

```
type
   PETableAliasNameType = array [0..PE_TABLE_NAME_LEN -1] of Char;
   PEDatabaseFieldNameType = array [0..PE_DATABASE_FIELD_NAME_LEN -1]
   of Char;
   PEReportFieldMappingInfo = record
   StructSize : Word;
   valueType : Word;
   tableAliasName : PETableAliasNameType;
   databaseFieldName : PEDatabaseFieldNameType;
   mappingTo : Integer;
end;
```

Компоненты записи	Описание компонентов
`StructSize`	Размер записи `PEReportFieldMappingInfo`. Инициализация выполняется с помощью константы `PE_SIZEOF_REPORT_FIELDMAPPING_INFO`
`valueType`	Индикатор типа поля: Константа `PE_FVT_INT8SFIELD` Описание — 8-битное целочисленное с символом Константа `PE_FVT_INT8UFIELD` Описание — 8-битное целочисленное без символа Константа `PE_FVT_INT16SFIELD` Описание — 16-битное целочисленное с символом Константа `PE_FVT_INT16UFIELD` Описание — 16-битное целочисленное без символа Константа `PE_FVT_INT32SFIELD` Описание — 32-битное целочисленное с символом Константа `PE_FVT_INT32UFIELD` Описание — 32-битное целочисленное без символа Константа `PE_FVT_NUMBERFIELD` Описание — цифровое поле Константа `PE_FVT_CURRENCYFIELD` Описание — денежная величина Константа `PE_FVT_BOOLEANFIELD` Описание — булева величина Константа `PE_FVT_DATEFIELD` Описание — дата Константа `PE_FVT_TIMEFIELD` Описание — время

(окончание)

Компоненты записи	Описание компонентов
valueType	Константа PE_FVT_STRINGFIELD Описание — строка Константа PE_FVT_TRANSIENTMEMOFIELD Описание — Transient Memo Константа PE_FVT_PERSISTENTMEMOFIELD Описание — Persistent Memo Константа PE_FVT_BLOBFIELD Описание — BLOB Константа PE_FVT_DATETIMEFIELD Описание — Дата/Время Константа PE_FVT_BITMAPFIELD Описание — картинка (BMP) Константа PE_FVT_ICONFIELD Описание — иконка Константа PE_FVT_PICTUREFIELD Описание — картинка Константа PE_FVT_OLEFIELD Описание — OLE-объект Константа PE_FVT_GRAPHFIELD Описание — графический объект Константа PE_FVT_UNKNOWNFIELD Описание — неизвестный тип данных
tableAliasName	Имя таблицы в базе данных. Длина имени определяется константой PE_TABLE_NAME_LEN = 128
databaseFieldName	Имя колонки в таблице. Длина имени определяется константой PE_DATABASE_FIELD_NAME_LEN = 128
mappingTo	Индекс поля в массиве databaseField, входящем в PEFieldMappingEventInfo. Если поле не определено, то его индекс равен -1

Переменная **PEReportAlertInfo** — содержит информацию о Report Alerts. Эта информация используется в функции PEGetNthReportAlert.

```
type PEReportAlertInfo = record
    StructSize : Word;
    nameLength : Smallint;
    name : HWnd;
    isEnabled : Smallint;
```

```
    alertConditionLength : Smallint;
    alertConditionFormula : HWnd;
    nTriggeredInstances : DWord;
    alertMessageLength : SmallInt;
    defaultAlertMessageLength : SmallInt;
    alertMessageFormula : HWnd;
    defaultAlertMessage : HWnd;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEReportAlertInfo. Инициализируется с помощью константы PE_SIZEOF_REPORT_ALERT_INFO
namelength	Длина имени Report Alert
name	Указатель на строку, содержащую имя Report Alert. Это имя присваивается в момент создания Report Alert
isEnabled	Определяет доступность Report Alert. Установите TRUE для обеспечения доступа или PE_UNCHANGED, если не требуется вносить изменений
alertConditionLength	Длина условия, определенного для Report Alert
alertConditionFormula	Указатель на строку, содержащую формулу Report Alert
nTriggeredInstances	Количество отработок Report Alert
alertMessageLength	Длина сообщения для Report Alert
defaultAlertMessageLength	Длина сообщения по умолчанию для Report Alert
alertMessageFormula	Указатель на формулу, используемую для создания сообщения Report Alert
defaultAlertMessge	Указатель на сообщение по умолчанию для Report Alert

Переменная **PEReportOptions** — содержит информацию о параметрах отчета. Эта информация используется в функции PEGetReportOptions для получения текущих параметров и в PESetReportOptions для их изменения.

```
type PEReportOptions = record
    StructSize : Word;
    saveDataWithReport : Smallint;
    saveSummariesWithReport : Smallint;
    useIndexForSpeed : Smallint;
    translateDOSStrings : Smallint;
    translateDOSMemos : Smallint;
```

```
    convertDateTimeType : Smallint;
    convertNullFieldToDefault : Smallint;
    morePrintEngineErrorMessages : Smallint;
    caseInsensitiveSQLData : Smallint;
    verifyOnEveryPrint : Smallint;
    zoomMode : Smallint;
    hasGroupTree : Smallint;
    dontGenerateDataForHiddenObjects : Smallint;
    performGroupingOnServer : Smallint;
    doAsyncQuery : Smallint;
    promptMode : Smallint;
    selectDistinctRecords : Smallint;
    alwaysSortLocally : Smallint;
    isReadOnly : Smallint;
    canSelectDistinctRecords : Smallint;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи PEReportOptions. Определяется через константу PE_SIZEOF_REPORT_OPTIONS
saveDataWithReport	Флаг, определяющий необходимость сохранения данных вместе с отчетом или PE_UNCHANGED, если не требуется вносить изменения
saveSummariesWithReport	Флаг, определяющий необходимость сохранения суммируемых данных вместе с отчетом или PE_UNCHANGED, если не требуется вносить изменения
useIndexForSpeed	Флаг, определяющий необходимость использования индексов на сервере для ускорения доступа к данным или PE_UNCHANGED для использования параметров по умолчанию
translateDOSStrings	Флаг, указывающий на необходимость преобразования строк в формате DOS или PE_UNCHANGED, если не требуется вносить изменения
translateDOSMemos	Флаг, указывающий на необходимость преобразования полей MEMO в формате DOS или PE_UNCHANGED, если не требуется вносить изменения

(продолжение)

Компоненты записи	Описание компонентов
convertDateTimeType	Флаг конвертации полей типа Дата/Время: Константа PE_RPTOPT_CVTDATETIMETOSTR Значение 0 Константа PE_RPTOPT_CVTDATETIMETODATE Значение 1 Константа PE_RPTOPT_KEEPDATETIMETYPE Значение 2
convertNullFieldToDefault	Флаг конвертации значений NULL в значения по умолчанию или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
morePrintEngineErrorMessages	Флаг показа дополнительных сообщений об ошибках или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
caseInsensitiveSQLData	Флаг отключения чувствительности к регистру букв или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
verifyOnEveryPrint	Флаг проверки отчета при каждой печати или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
zoomMode	Используется одна из констант PE_ZOOM_XXX или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
hasGroupTree	Флаг показа дерева групп при просмотре отчета или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
dontGenerateDataForHiddenObjects	Флаг генерации данных для скрытых объектов или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
performGroupingOnServer	Флаг выполнения группировки на сервере или PE_UNCHANGED, если не требуется изменять ранее определенные параметры

(окончание)

Компоненты записи	Описание компонентов
doAsyncQuery	Флаг выполнения запросов асинхронно или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
promptMode	Флаг запроса ввода параметров при запуске отчета на обработку или PE_UNCHANGED, если не требуется изменять ранее определенные параметры: Константа PE_RPTOPT_PROMPT_NONE Значение 0 Константа PE_RPTOPT_PROMPT_NORMAL Значение 1 Константа PE_RPTOPT_PROMPT_ALWAYS Значение 2
selectDistinctRecords	Флаг выборки данных без учета дубликатов или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
alwaysSortLocally	Флаг, позволяющий выполнять сортировку данных локально или PE_UNCHANGED, если не требуется изменять ранее определенные параметры
isReadOnly	Флаг, указывающий на то, что отчет может быть использован только для чтения. Данное свойство только для чтения
canSelectDistinctRecords	Указатель того, что данные в отчет могут выбираться без дублирования. Данное свойство только для чтения

Замечание

Каждый элемент записи может быть определен с помощью TRUE, FALSE или PE_UNCHANGED в случае, если не требуется вносить изменений.

Переменная **PEReportSummaryInfo** — содержит суммарную информацию об отчете.

```
type
   PEApplicationNameType = array[0..PE_SI_APPLICATION_NAME_LEN-1]
   of Char;
   PETitleType = array[0..PE_SI_TITLE_LEN-1] of Char;
```

```
    PESubjectType = array[0..PE_SI_SUBJECT_LEN-1] of Char;
    PEAuthorType = array[0..PE_SI_AUTHOR_LEN-1] of Char;
    PEKeywordsType = array[0..PE_SI_KEYWORDS_LEN-1] of Char;
    PECommentsType = array[0..PE_SI_COMMENTS_LEN-1] of Char;
    PEReportTemplate = array[0..PE_SI_REPORT_TEMPLATE_LEN-1] of Char;
    PEReportSummary = record
    StructSize: Integer;
    applicationName: PEApplicationNameType;
    title: PETitleType;
    subject: PESubjectType;
    author: PEAuthorType;
    keywords: PEKeywordsType;
    comments: PEReportTemplate;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи. Инициализируется с помощью константы PE_SIZEOF_REPORT_SUMMARY_INFO
applicationName	Имя приложения, использующего отчет. Длина имени определяется PE_APPLICATION_NAME_LEN = 128. Этот параметр только для чтения
title	Заголовок отчета. Длина заголовка задается константой PE_TITLE_LEN = 128
subject	Тема отчета. Длина текста задается константой PE_SI_SUBJECT_LEN = 128
author	Автор отчета. Длина текста задается константой PE_SI_AUTHOR_LEN = 128
keywords	Ключевые слова, включенные в отчет. Длина текста задается константой PE_SI_KEYWORDS_LEN = 128
comments	Комментарии к отчету. Длина текста задается константой PE_SI_COMMENTS_LEN = 512
reportTemplate	Шаблон отчета. Длина текста задается константой PE_SI_REPORT_TEMPLATE_LEN = 128
savePreviewPicture	Флаг, указывающий на необходимость сохранения "снимка" отчета или PE_UNCHANGED, если требуется использовать ранее заданные параметры

Переменная **PESearchButtonClickedEventInfo** — содержит информацию о нажатии кнопки поиска в окне просмотра отчета.

```
type
    PESearchStringType = array[0..PE_SEARCH_STRING_LEN-1] of Char;
    PESearchButtonClickedEventInfo = record
    windowHandle: Longint;
```

```
    searchString: PESearchStringType;
    StructSize: Word;
end;
```

Компоненты записи	Описание компонентов
windowHandle	Указатель на окно, в котором произошло событие
searchString	Строка поиска. Длина текста задается константой PE_SEARCH_STRING_LEN = 128
StructSize	Размер записи PESearchButtonClickedEventInfo. Инициализируется с помощью константы PE_SIZEOF_SEARCH_BUTTON_CLICKED_EVENT_INFO

Переменная **PESectionOptions** — содержит спецификацию форматирования секции или области отчета. Эти данные используются в функциях PEGetSectionFormat, PEGetAreaFormat, PESetSectionFormat и PESetAreaFormat.

```
type PESectionOptions = record
    StructSize: Word;
    visible: Smallint;
    newPageBefore: Smallint;
    newPageAfter: Smallint;
    keepTogether: Smallint;
    suppressBlankSection: Smallint;
    resetPageNAfter: Smallint;
    printAtBottomOfPage: Smallint;
    backgroundColor: COLORREF;
    underlaySection: Smallint;
    showArea: Smallint;
    freeFormPlacement: Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PESectionOptions. Определяется через константу PE_SIZEOF_SECTION_OPTIONS
visible	Флаг, указывающий на видимость/невидимость секции
newPageBefore	Флаг, указывающий на необходимость формирования разрыва страницы перед печатью секции
newPageAfter	Флаг, указывающий на необходимость формирования разрыва страницы после печати секции

(окончание)

Компоненты записи	Описание компонентов
keepTogether	Флаг, указывающий на необходимость совместного размещения объектов из одной секции
suppressBlankSection	Флаг, позволяющий скрыть пустую секцию
resetPageNAfter	Флаг, обеспечивающий возможность переустановки номера страницы после формирования секции
printAtBottomOfPage	Флаг, предоставляющий возможность печатать информацию только внизу страницы
backgroundColor	Цвет RGB, описанный в COLORREF или PE_NO_COLOR, если цвет не нужен
underlaySection	Индикатор подчеркивания секции
showArea	Флаг показа или скрытия раздела
freeFormPlacement	Флаг, используемый только на дизайне. Позволяет разместить объект в любом месте секции
reserveMinimumPageFooter	Флаг, позволяющий уменьшить пустое место в нижней части страницы. Если равен TRUE, то секция в нижней части будет автоматически минимизирована; если равен FALSE (значение по умолчанию), то секция будет занимать столько места, сколько было зарезервировано в момент разработки отчета

Переменная **PESessionInfo** — содержит информацию о текущей сессии Microsoft Access. Многие таблицы базы данных Microsoft Access требуют того, чтобы перед использованием данных была открыта сессия. Используется в функциях PEGetNthTableSessionInfo и PESetNthTableSessionInfo.

```
type
   PESesPassType = array[1..PE_SESS_PASSWORD_LEN] of Char;
   PESessionInfo = record
   StructSize: Word;
   UserID: PESesPassType;
   Password: PESesPassType;
   SessionHandle: DWord;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PESessionInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_SESSION_INFO

(окончание)

Компоненты записи	Описание компонентов
UserID	Идентификатор пользователя. Длина строки определяется константой PE_SESS_USERID_LEN = 128
Password	Пароль пользователя. Длина строки определяется константой PE_SESS_PASSWORD_LEN = 128. Пароль недоступен в случае получения информации из отчета
SessionHandle	Указатель на текущую сессию MS Access

Переменная **PEShowGroupEventInfo** — содержит информацию в том случае, когда функция обратной связи вызывается по событию с идентификатором ID = PE_SHOW_GROUP_EVENT.

```
type
   PEPCharPointer = PChar;
   PEShowGroupEventInfo = record
   StructSize: Word;
   groupLevel: Word;
   windowHandle: Longint;
   groupList: PEPCharPointer;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEShowGroupEventInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_SHOW_GROUP_EVENT_INFO
groupLevel	Номер группы в списке условий группировки
windowHandle	Указатель на окно просмотра отчета, в котором произошло событие
groupList	Массив имен групп, по которым выполняется навигация

Переменная **PEStartEventInfo** — содержит начальную информацию о событии в том случае, когда функция обратной связи отработала по идентификатору события ID = PE_START_EVENT.

```
type PEStartEventInfo = record
   StructSize: Word;
   destination: Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEStartEventInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_START_EVENT_INFO
destination	Назначение процесса: Константа PE_TO_NOWHERE Описание — нет назначения Константа PE_TO_WINDOW Описание — просмотр отчета Константа PE_TO_PRINTER Описание — печать отчета на принтер Константа PE_TO_EXPORT Описание — экспорт отчета Константа PE_FROM_QUERY Описание — обработка отчета через запрос

Переменная **PEStopEventInfo** — содержит информацию о завершении события в том случае, когда функция обратной связи отработала по идентификатору события ID = PE_STOP_EVENT.

```
type PEStopEventInfo  = record
    StructSize: Word;
    destination: Word;
    jobStatus: Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEStopEventInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_STOP_EVENT_INFO
destination	Назначение процесса: Константа PE_TO_NOWHERE Описание — нет назначения Константа PE_TO_WINDOW Описание — просмотр отчета Константа PE_TO_PRINTER Описание — печать отчета на принтер Константа PE_TO_EXPORT Описание — экспорт отчета Константа PE_FROM_QUERY Описание — обработка отчета через запрос

(окончание)

Компоненты записи	Описание компонентов
jobStatus	Константа, показывающая текущий статус процесса: Константа PE_TO_NOWHERE Описание — нет назначения Константа PE_TO_WINDOW Описание — просмотр отчета Константа PE_TO_PRINTER Описание — печать отчета на принтер Константа PE_TO_EXPORT Описание — экспорт отчета Константа PE_FROM_QUERY Описание — обработка отчета через запрос

Переменная **PESubreportInfo** — содержит информацию о подотчете. Эта запись используется в функции PEGetSubreportInfo для сбора информации о выбранном подотчете.

```
type
   PESubreportNameType = array[0..PE_SUBREPORT_NAME_LEN-1] of Char;
   PESubreportInfo = record
   structSize : Word;
   name : PESubreportNameType;
   NLinks : Smallint;
   IsOnDemand : Smallint;
   external : Bool;
   reimportOption : Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PESubreportInfo. Инициализация данного параметра выполняется с помощью константы PE_SIZEOF_SUBREPORT_INFO
name	Массив, содержащий имена подотчетов. Длина имени подотчета определяется константой PE_SUBREPORT_NAME_LEN = 128
NLinks	Номер связи с данными в основном отчете
isOnDemand	TRUE — если подотчет отображается совместно с основным отчетом. FALSE — если подотчет отображается по требованию

(окончание)

Компоненты записи	Описание компонентов
external	TRUE — если подотчет импортирован. FALSE — если встроен
reimportOption	Определяет параметры обновления данных в подотчете. Равен PE_SRI_ONOPENJOB, когда требуется обновить данные, когда открывается главный отчет, или PE_SRI_ONFUNCTIONCALL, т. е. обновить данные только в случае вызова через функции API

Переменная **PETableDifferenceInfo** — запись, используемая только для чтения. Содержит информацию о таблице базы данных и используется в функции PECheckNthTableDifferences.

```
type PETableDifferenceInfo = record
    StructSize       : Word;
    tableDifferences : DWord;
    reserved1        : DWord;
    reserved2        : DWord;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PETableLocation. Определяется через константу PE_SIZEOF_TABLE_DIFFERENCE_INFO
tableDifferences	Только для чтения. Возвращает комбинацию констант PE_TCD_XXX: Константа PE_TCD_OKAY Значение — 0x00000000 Константа PE_TCD_DATABASENOTFOUND Значение — 0x00000001 Константа PE_TCD_SERVERNOTFOUND Значение — 0x00000002 Константа PE_TCD_SERVERNOTOPENED Значение — 0x00000004 Константа PE_TCD_ALIASCHANGED Значение — 0x00000008 Константа PE_TCD_INDEXESCHANGED Значение — 0x00000010 Константа PE_TCD_DRIVERCHANGED Значение — 0x00000020

(продолжение)

Компоненты записи	Описание компонентов
tableDifferences	Константа PE_TCD_DICTIONARYCHANGED Значение — 0x00000040 Константа PE_TCD_FILETYPECHANGED Значение — 0x00000080 Константа PE_TCD_RECORDSIZECHANGED Значение — 0x00000100 Константа PE_TCD_ACCESSCHANGED Значение — 0x00000200 Константа PE_TCD_PARAMETERSCHANGED Значение — 0x00000400 Константа PE_TCD_LOCATIONCHANGED Значение — 0x00000800 Константа PE_TCD_DATABASEOTHER Значение — 0x00001000 Константа PE_TCD_NUMFIELDSCHANGED Значение — 0x00010000 Константа PE_TCD_FIELDOTHER Значение — 0x00020000 Константа PE_TCD_FIELDNAMECHANGED Значение — 0x00040000 Константа PE_TCD_FIELDDESCCHANGED Значение — 0x00080000 Константа PE_TCD_FIELDTYPECHANGED Значение — 0x00100000 Константа PE_TCD_FIELDSIZECHANGED Значение — 0x00200000 Константа PE_TCD_NATIVEFIELDTYPECHANGED Значение — 0x00400000 Константа PE_TCD_NATIVEFIELDOFFSETCHANGED Значение — 0x00800000 Константа PE_TCD_NATIVEFIELDSIZECHANGED Значение — 0x01000000 Константа PE_TCD_FIELDDECPLACESCHANGED Значение — 0x02000000
reserved1	Зарезервировано. Не используется
reserved2	Зарезервировано. Не используется

Переменная **PETableLocation** — содержит информацию о местоположении базы данных и используется в функциях PEGetNthTableLocation и PESetNthTableLocation.

```
type
   PETableLocType = array [0..PE_TABLE_LOCATION_LEN - 1] of Char;
   PEConnectBufferType = array [0..PE_CONNECTION_BUFFER_LEN - 1]
   of Char;
   PETableLocation = record
   StructSize    : Word;
   Location      : PETableLocType;
   SubLocation   : PETableLocType; { For MS Access Table Names }
   ConnectBuffer : PEConnectBufferType;
end;
```

Компоненты записи	**Описание компонентов**
StructSize	Размер записи PETableLocation. Размер задается константой PE_SIZEOF_TABLE_LOCATION
Location	Массив с информацией о местоположении баз данных. Длина строки массива задается константой PE_TABLE_LOCATION_LEN = 256. Данное значение должно быть отформатировано в соответствии с представленными примерами, иначе возможны проблемы при работе с функциями, использующими данную запись: • xBASE (Natively): <drive>:\<path>\<file>; • xBASE (ODBC): <datasource name>; • Paradox (Natively): <drive>:\<path>\<file>; • Paradox (ODBC): <datasource name>; • Btrieve (Natively): <drive>:\<path>\<file>; • Btrieve (ODBC): <datasource name>; • Oracle (Natively): <database>.<table>; • Oracle (ODBC): <database>.<table>; • SQL Server (Natively): <database>.<owner>.<table>; • SQL Server (ODBC): <database>.<owner>.<table>
SubLocation	Дополнительные варианты местоположения базы данных. Длина строки массива задается константой PE_TABLE_LOCATION_LEN = 256
ConnectBuffer	Массив буферов, содержащих информацию о подключенных таблицах. Длина буфера определяется константой PE_CONNECTION_BUFFER_LEN = 512

Переменная **PETablePrivateInfo** — содержит информацию об использовании объектов данных, таких как ADO, DAO, RDO или CDO, определенных через Active Data Drivers (crdb_ado.dll, crdb_dao.dll, crdb_odbc.dll, crdb_cdo.dll).

```
type
    crBytePointer = Byte;
    PETablePrivateInfo = record
    StructSize: Word;
    nBytes: Smallint;
    tag: DWord;
    dataPtr: crBytePointer;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PETablePrivateInfo. Определяется с помощью константы PE_SIZEOF_TABLE_PRIVATE_INFO
nBytes	Длина строки dataPtr
tag	На данный момент значение должно быть равно 3 для всех вариантов объектов данных
dataPtr	Указатель на различные варианты драйверов. Для DAO, ADO или элементов Visual Basic With — Recordset, для CDO — Rowset

Переменная **PETableType** — содержит информацию, необходимую для идентификации типа указанной таблицы. Эта информация собирается с помощью функции PEGetNthTableType.

```
type
    PEDllNameType = array[0..PE_DLL_NAME_LEN-1] of Char;
    PEFullNameType = array[0..PE_FULL_NAME_LEN-1] of Char;
    PETableType = record
    StructSize: Word;
    DLLName: PEDllNameType;
    DescriptiveName: PEFullNameType;
    DBType: Word;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PETableType. Установка данного параметра выполняется с помощью константы PE_SIZEOF_TABLE_TYPE

(окончание)

Компоненты записи	Описание компонентов
DLLName	Имя библиотеки, обеспечивающей связь с базой данных для интересующей таблицы. Длина имени определяется константой PE_DLL_NAME_LEN = 64 Библиотека — crdb_p2bbde.dll База данных — Borland Database Engine Библиотека — crdb_dao.dll База данных — DAO (Access) Библиотека — crdb_p2bbde.dll База данных — Paradox Библиотека — crdb_p2bxbse.dll База данных — dBASE, FoxPro, Clipper Библиотека — crdb_p2bbtrv.dll База данных — Btrieve Библиотека — crdb_p2sdb2.dll База данных — DB2/2 Библиотека — crdb_odbc.dll База данных — ODBC Библиотека — crdb_oracle.dll База данных — Oracle Библиотека — crdb_p2ssyb10.dll База данных — Sybase 10/11
DescriptiveName	Полное описание таблицы. Длина описания определяется константой PE_FULL_NAME_LEN = 256
DBType	Тип базы данных, которая содержит указанную таблицу Константа PE_DT_STANDARD Описание — стандартная база данных (не SQL) Константа PE_DT_SQL Описание — реляционная база данных Константа PE_DT_SQL_STORED_PROCEDURE Описание — хранимая процедура

Переменная **PETrackCursorInfo** — содержит информацию о виде курсора, отображаемого в окне просмотра отчета. Используется в функциях PEGetTrackCursorInfo и PESetTrackCursorInfo.

```
type PETrackCursorInfo = record
    StructSize: Word;
    groupAreaCursor: Smallint;
```

```
    groupAreaFieldCursor: Smallint;
    detailAreaCursor: Smallint;
    detailAreaFieldCursor: Smallint;
    graphCursor: Smallint;
    groupAreaCursorHandle: Longint;
    groupAreaFieldCursorHandle: Longint;
    detailAreaCursorHandle: Longint;
    detailAreaFieldCursorHandle: Longint;
    graphCursorHandle: Longint;
    ondemandSubreportCursor: Smallint;
    hyperlinkCursor: Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PETrackCursorInfo. Установка данного параметра выполняется с помощью константы PE_SIZEOF_TRACK_CURSOR_INFO
groupAreaCursor	Определяет тип курсора, отображаемый в разделе группировки. Используется одна из констант PE_TC_XXX или PE_UNCHANGED, если необходимо оставить курсор без изменений
groupAreaFieldCursor	Определяет тип курсора, отображаемый в разделе группировки у различных полей. Используется одна из констант PE_TC_XXX или PE_UNCHANGED, если необходимо оставить курсор без изменений
detailAreaCursor	Определяет тип курсора, отображаемый в детальном разделе отчета. Используется одна из констант PE_TC_XXX или PE_UNCHANGED, если необходимо оставить курсор без изменений
detailAreaFieldCursor	Определяет тип курсора, отображаемый в детальном разделе отчета у различных полей. Используется одна из констант PE_TC_XXX или PE_UNCHANGED, если необходимо оставить курсор без изменений
graphCursor	Определяет тип курсора, отображаемый у графика, находящегося в разделах Report Header или Report Footer. Используется одна из констант PE_TC_XXX или PE_UNCHANGED, если необходимо оставить курсор без изменений
groupAreaCursorHandle	Зарезервировано. Не используется
groupAreaFiedlCursorHandle	Зарезервировано. Не используется
detailAreaCursorHandle	Зарезервировано. Не используется
detailAreaFieldCursorHandle	Зарезервировано. Не используется

(окончание)

Компоненты записи	Описание компонентов
graphCursorHandle	Зарезервировано. Не используется
ondemandSubreportCursor	Определяет курсор, отображаемый для подотчета, вызываемого по требованию. По умолчанию используется PE_TC_MAGNIFY_CURSOR
hyperlinkCursor	Определяет тип курсора, отображаемый для гиперссылок. По умолчанию используется PE_TC_HAND_CURSOR

Переменная **PEValueInfo** — содержит информацию, используемую в функциях PEConvertPFInfotoVInfo, PEConvertVInfotoPFInfo и PESetNthParameterField.

```
type
   PEVALUEINFOSTRINGTYPE = array[0..PE_VI_STRING_LEN-1] of Smallint;
   PEVALUEINFODATEORTIMETYPE = array[0..5] of Smallint;
   PEVALUEINFODATETIMETYPE = array[0..2] of Smallint;
   PEValueInfo = record
   StructSize: Word;
   valueType: Word;
   viNumber: Double;
   viCurrency: Double;
   viBoolean: Bool;
   viString: PEVALUEINFOSTRINGTYPE;
   viDate: PEVALUEINFODATEORTIMETYPE;
   viDateTime: PEVALUEINFODATETIMETYPE;
   viTime: PEVALUEINFODATEORTIMETYPE;
   viColor: COLORREF;
   viInteger: Smallint;
   viC: Char;
   ignored: Char;
   viLong: Longint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEValueInfo. Установка данного параметра выполняется с помощью константы PE_SIZEOF_VALUE_INFO
valueType	Тип данных поля-параметра: Тип данных Number Константа PE_VI_NUMBER Тип данных Currency Константа PE_VI_CURRENCY

(окончание)

Компоненты записи	Описание компонентов
`valueType`	Тип данных Boolean Константа `PE_VI_BOOLEAN` Тип данных Date Константа `PE_VI_DATE` Тип данных String Константа `PE_VI_STRING` Тип данных DateTime Константа `PE_VI_DATETIME` Тип данных Time Константа `PE_VI_TIME` Тип данных Integer Константа `PE_VI_INTEGER` Тип данных COLOREF Константа `PE_VI_COLOR` Тип данных Char Константа `PE_VI_CHAR` Тип данных Long Константа `PE_VI_LONG` Тип данных No Value Константа `PE_VI_NOVALUE`
`ViNumber`	Содержит значение, если параметр является цифровым
`viCurrency`	Содержит значение, если параметр является денежным
`viBoolean`	Содержит значение, если параметр является булевым
`ViString`	Содержит значение, если параметр является строковым
`ViDate`	Содержит значение, если параметр является датой (год, месяц, число)
`viDateTime`	Содержит значение, если параметр является Датой/Временем (год, месяц, число, часы, минуты, секунды)
`ViTime`	Содержит значение, если параметр является временем (часы, минуты, секунды)
`ViColor`	Содержит значение, если параметр определяет цвет
`viInteger`	Содержит значение, если параметр является целочисленным
`ViC`	Содержит значение, если параметр является символьным
`Ignored`	Только для внутреннего использования. Не используется
`ViLong`	Содержит значение, если параметр относится к типу Long...

Переменная **PEVersionInfo** — содержит информацию о версии отчета. Эта информация используется в функции PEGetReportVersion.

```
type PEVersionInfo = record
    StructSize : Word;
    major : Word;
    minor : Word;
    character : PChar;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEVersionInfo. Установка данного параметра выполняется с помощью константы PE_SIZEOF_VERSION_INFO
major	Определяет наивысшую версию
minor	Определяет наименьшую версию
letter	Определяет символьную часть низшей версии. Большинство отчетов не имеет символьной составляющей, поэтому возвращается значение NULL

Переменная **PEWindowOptions** — содержит информацию, связанную с окном просмотра. Эта запись используется в функциях PEGetWindowOptions и PESetWindowOptions.

```
type PEWindowOptions = record
    StructSize: Word;
    hasGroupTree: Smallint;
    canDrillDown: Smallint;
    hasNavigationControls: Smallint;
    hasCancelButton: Smallint;
    hasPrintButton: Smallint;
    hasExportButton: Smallint;
    hasZoomControl: Smallint;
    hasCloseButton: Smallint;
    hasProgressControls: Smallint;
    hasSearchButton: Smallint;
    hasPrintSetupButton: Smallint;
    hasRefreshButton: Smallint;
    showToolbarTips: Smallint;
    showDocumentTips: Smallint;
    hasLaunchButton: Smallint;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEWindowOptions. Установка данного параметра выполняется с помощью константы PE_SIZEOF_WINDOW_OPTIONS
hasGroupTree	Определяет необходимость отображения дерева групп в окне просмотра (TRUE/FALSE)
canDrillDown	Определяет необходимость разрешить выполнение специальной выборки данных в окне просмотра (TRUE/FALSE)
hasNavigationControls	Определяет необходимость отображения кнопок навигации в окне просмотра (TRUE/FALSE)
hasCancelButton	Определяет необходимость отображения кнопки отмены в окне просмотра (TRUE/FALSE)
hasPrintButton	Определяет необходимость отображения кнопки печати в окне просмотра (TRUE/FALSE). По умолчанию это значение равно TRUE.
hasExportButton	Определяет необходимость отображения кнопки экспорта в окне просмотра (TRUE/FALSE). По умолчанию это значение равно TRUE
hasZoomControl	Определяет необходимость отображения элементов управления размерами в окне просмотра (TRUE/FALSE). По умолчанию это значение равно TRUE
hasCloseButton	Определяет необходимость отображения кнопки закрытия окна просмотра (TRUE/FALSE). По умолчанию это значение равно FALSE
hasProgressControls	Определяет необходимость отображения процесса загрузки данных в окне просмотра (TRUE/FALSE). По умолчанию это значение равно TRUE
hasSearchButton	Определяет необходимость отображения кнопки поиска в окне просмотра (TRUE/FALSE). По умолчанию это значение равно FALSE
hasPrintSetupButton	Определяет необходимость отображения кнопки настройки принтера в окне просмотра (TRUE/FALSE)
hasRefreshButton	Определяет необходимость отображения кнопки обновления данных в окне просмотра (TRUE/FALSE)
showToolbarTips	Определяет необходимость отображения подсказок у кнопок в окне просмотра (TRUE/FALSE). По умолчанию это значение равно TRUE
showDocumentTips	Определяет необходимость отображения подсказок для элементов отчета в окне просмотра (TRUE/FALSE). По умолчанию это значение равно FALSE
hasLaunchButton	Определяет необходимость отображения кнопки ссылки на Crystal Analysis в окне просмотра (TRUE/FALSE)

Переменная **PEZoomLevelChangingEventInfo** — содержит информацию о записях, считанных в момент срабатывания функции обратной связи по событию с идентификатором ID = PE_ZOOM_LEVEL_CHANGING_EVENT.

```
type PEZoomLevelChangingEventInfo = record
    StructSize: Word;
    zoomLevel: Word;
    windowHandle: HWnd;
end;
```

Компоненты записи	Описание компонентов
StructSize	Размер записи PEZoomLevelChangingEventInfo. Установка данного параметра выполняется с помощью константы PE_SIZEOF_ZOOM_LEVEL_CHANGNG_EVENT_INFO
zoomLevel	Уровень увеличения отчета в окне просмотра. Должен попадать в диапазон от 25 до 400 либо можно воспользоваться константами: PE_ZOOM_FULL_SIZE PE_ZOOM_SIZE_FIT_ONE_SIDE PE_ZOOM_SIZE_FIT_BOTH_SIDES
windowHandle	Указатель окна, в котором выполняется изменение масштаба отчета

Переменная **UXDDiskOptions** — содержит информацию, которая используется в записи PEEnableEventInfo при экспорте.

```
type UXDDiskOptions = record
    structSize: Word;
    fileName: PChar;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXDDiskOptions. Установка данного параметра выполняется с помощью константы UXDDiskOptionsSize
fileName	Строка, которая содержит полное имя файла для сохранения

Переменная **UXDMAPIOptions** — содержит информацию об электронной почте, которая используется в записи PEEnableEventInfo для экспорта с использованием MAPI.

```
type UXDMAPIOptions = record
    structSize: Word;
```

```
    toList: PChar;
    ccList: PChar;
    subject: PChar;
    message: PChar;
    nRecipients: Word;
    recipients: lpMapiRecipDesc;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXDMAPIOptions. Установка данного параметра выполняется с помощью константы UXDMAPIOptionsSize
toList	Строка, содержащая список адресатов. Если установлено значение для параметра recipients, то данный параметр игнорируется
ccList	Строка, содержащая список адресатов, которым отправляется копия. Если установлено значение для параметра recipients, то данный параметр игнорируется
subject	Заголовок письма
message	Текст письма
nRecipients	Количество получателей данного сообщения. Необходимо установить 0 в случае когда параметры toList и ccList заполнены; только для языка С
recipients	Массив, который содержит информацию о получателях сообщения. Для получения более полной информации необходимо обратиться к Microsoft's MAPI documentation; только для языка С

Переменная **UXDSMIOptions** — содержит информацию об электронной почте, которая используется в записи PEEnableEventInfo для экспорта с использованием MAPI.

```
type UXDSMIOptions = record
    structSize: Word;
    toList: PChar;
    ccList: PChar;
    subject: PChar;
    message: PChar;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXDMAPIOptions. Установка данного параметра выполняется с помощью константы UXDMAPIOptionsSize

(окончание)

Компоненты записи	Описание компонентов
toList	Строка, содержащая список адресатов. Если установлено значение для параметра recipients, то данный параметр игнорируется
ccList	Строка, содержащая список адресатов, которым отправляется копия. Если установлено значение для параметра recipients, то данный параметр игнорируется
subject	Заголовок письма
message	Текст письма

Переменная **UXDPostFolderOptions** — содержит информацию, необходимую для экспорта в Microsoft Exchange. Используется в записи PEEnableEventInfo.

```
type UXDPostFolderOptions = record
    structSize: Word;
    pszProfile: PChar;
    pszPassword: PChar;
    wDestType: Word;
    pszFolderPath: PChar;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXDPostFolderOptions. Установка данного параметра выполняется с помощью константы UXDPostFolderOptionsSize
pszProfile	Профиль Exchange
pszPassword	Пароль для Exchange
wDestType	Тип экспорта отчета: Константа UXDExchFolderType Значение — 0 Описание — wDestType for Microsoft Exchange folder Константа UXDPostDocMessage Значение — 1009 Описание — wDestType for folder messages Константа UXDPostPersonalReport Значение — 1010 Описание — wDestType for personal report Константа UXDPostFolderReport Значение — 1011 Описание — wDestType for folder report

(окончание)

Компоненты записи	Описание компонентов
pszFolderPath	Путь к папке Exchange, в которую программа поместит экспортируемый файл. Параметр pszFolderPath должен иметь следующий формат: <Message Store Name>@<Folder Name>@<Folder Name>

Переменная **UXFCharSeparatedOptions** — содержит информацию о том, как выполнять экспорт цифр и дат из отчета в формат Character Separated.

```
type UXFCharSeparatedOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
    stringDelimiter: Char;
    fieldDelimiter: PChar;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXFCharSeparatedOptions. Установка данного параметра выполняется с помощью константы UXFCharSeparatedOptionsSize
useReportNumberFormat	Если равен TRUE, то в момент экспорта будет использован тот же формат для цифр, что и в отчете; если равен FALSE, то формат цифр будет оптимизирован под выбранный формат файла
useReportDateFormat	Если равен TRUE, то в момент экспорта будет использован тот же формат для дат, что и в отчете; если равен FALSE, то формат дат будет оптимизирован под выбранный формат файла
stringDelimiter	Символ, который желательно использовать для разделения цифробуквенных полей. Можно использовать любой символ. Символ должен быть обязательно заключен в кавычки
fieldDelimiter	Строка, содержащая набор символов, которая будет использована для разделения полей. Количество символов в строке может достигать 16, но строка обязательно должна быть заключена в кавычки

Переменная **UXFCommaTabSeparatedOptions** — содержит информацию о том, как выполнять экспорт цифр и дат из отчета в форматы Comma-Separated или Tab-Separated.

```
type UXFCommaTabSeparatedOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXFCommaTabSeparatedOptions. Установка данного параметра выполняется с помощью константы UXFCommaTabSeparatedOptionsSize
useReportNumberFormat	Если равен TRUE, то в момент экспорта будет использован тот же формат для цифр, что и в отчете; если равен FALSE, то формат цифр будет оптимизирован под выбранный формат файла
useReportDateFormat	Если равен TRUE, то в момент экспорта будет использован тот же формат для дат, что и в отчете; если равен FALSE, то формат дат будет оптимизирован под выбранный формат файла

Переменная **UXFDIFOptions** — содержит информацию о том, как выполнять экспорт цифр и дат из отчета в формат DIF (Data Interchange Format).

```
type UXFDIFOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXFDIFOptions. Установка данного параметра выполняется с помощью константы UXFDIFOptionsSize
useReportNumberFormat	Если равен TRUE, то в момент экспорта будет использован тот же формат для цифр, что и в отчете; если равен FALSE, то формат цифр будет оптимизирован под выбранный формат файла
useReportDateFormat	Если равен TRUE, то в момент экспорта будет использован тот же формат для дат, что и в отчете; если равен FALSE, то формат дат будет оптимизирован под выбранный формат файла

Переменная **UXFHTML3Options** — содержит параметры, используемые в записи PEEnableEventInfo для экспорта отчета в формат HTML.

```
type UXFHTML3Options = record
    structSize: Word;
    fileName: PChar;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXFHTML3Options. Установка данного параметра выполняется с помощью константы UXFHTML3OptionsSize
fileName	Полное имя файла, например, "C:\pub\docs\boxoffic\default.htm". В случае экспорта графических файлов они сохраняются в тот же самый каталог

Переменная **UXFODBCOptions** — содержит информацию, необходимую при экспорте с использованием ODBC.

```
type UXFODBCOptions = record
    structSize: Word;
    dataSourceName: PChar;
    dataSourceUserID: PChar;
    dataSourcePassword: PChar;
    exportTableName: PChar;
end;
```

Компоненты записи	Описание компонентов
structSize	Размер записи UXFODBCOptions. Установка данного параметра выполняется с помощью константы UXFODBCOptionsSize
dataSourceName	Имя ODBC-источника, используемого для экспорта
dataSourceUserID	Имя пользователя, если необходимо
dataSourcePassword	Пароль
exportTableName	Имя таблицы, в которую будут сохранены экспортируемые данные

Переменная **UXFPaginatedTextOptions** — содержит информацию, используемую при экспорте в разбитый на страницы текст.

```
type UXFPaginatedTextOptions = record
    structSize: Word;
    nLinesPerPage: Word;
end;
```

Компоненты записи	**Описание компонентов**
`structSize`	Размер записи `UXFPaginatedTextOptions`. Установка данного параметра выполняется с помощью константы `UXFPaginatedTextOptionsSize`
`nLinesPerPage`	Количество строк на страницу. По умолчанию количество строк 60. Когда выполняется экспорт в данный формат, пользователю предлагается диалог для ввода количества строк на страницу

Переменная **`UXFRecordStyleOptions`** — содержит информацию о том, как выполнять экспорт цифр и дат из отчета в формат `Record style (columns of values)`.

```
type UXFRecordStyleOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
end;
```

Компоненты записи	**Описание компонентов**
`structSize`	Размер записи `UXFRecordStyleOptions`. Установка данного параметра выполняется с помощью константы `UXFRecordStyleOptionsSize`
`useReportNumberFormat`	Если равен `TRUE`, то в момент экспорта будет использован тот же формат для цифр, что и в отчете; если равен `FALSE`, то формат цифр будет оптимизирован под выбранный формат файла
`useReportDateFormat`	Если равен `TRUE`, то в момент экспорта будет использован тот же формат для дат, что и в отчете; если равен `FALSE`, то формат дат будет оптимизирован под выбранный формат файла